Signal Processing 100 (2014) 197–207

Contents lists available at ScienceDirect

Signal Processing

journal homepage: www.elsevier.com/locate/sigpro

Image encryption using binary bitplane

Yicong Zhou*, Weijia Cao, C.L. Philip Chen

Department of Computer and Information Science, University of Macau, Macau, China

ARTICLE INFO

Article history:
Received 6 November 2013
Received in revised form
19 January 2014
Accepted 21 January 2014
Available online 30 January 2014

Keywords:
Bitplane decomposition
Image encryption
Bit-level permutation
Security analysis

ABSTRACT

To enhance security of the bitplane decomposition based image encryption methods, this paper introduces a novel image encryption algorithm using a bitplane of a source image as the security key bitplane to encrypt images. Users have the flexibility to choose (1) any existing or newly generated image as the source image; (2) any decomposition method for generating the bitplane; (3) any decomposed bitplane as the security key bitplane; (4) any scrambling method for the bit-level permutation. As an example, this paper also proposes a bit-level scrambling algorithm to change bit positions. Simulations and security analysis are provided to demonstrate an excellent encryption performance of the proposed algorithm.

1. Introduction

With the rapid advance in network technologies and smartphone systems, a growing number of images and videos with private and confidential information are generated, transmitted and stored every minute. The increasing use of images and videos leads to the problem of how to prevent them from different types of security attacks and information leakage and to ensure the integrity of images and videos. Therefore, the security of images and videos has drawn researchers' attentions.

Many image encryption algorithms have been developed to protect images and videos in different fields [1–3]. Examples include traditional data encryption standards such as data encryption standard (DES) [4] and advanced encryption standard (AES) [5], and image encryption methods based on different technologies such as chaotic systems [6–8], random grids [9], wave transmission [10] and optical encryption techniques [11,12].

Among existing image encryption technologies, an interesting technique based on image bitplane decomposition has shown excellent encryption performance and is easily implemented in hardware [13]. This technique first decomposes an image into several binary bitplanes using a specific decomposition method, such as traditional binary decomposition, gray code decomposition [14] and Fibonacci p-code decomposition [15–17]. It then manipulates bitplanes using various methods, and combines bitplanes to obtain the encrypted images. Based on this technique, a list of image encryption algorithms has been proposed recently. Han et al. encrypted bitplanes of optical images using the exclusive-OR (XOR) operations [18]. Later, selective bitplane encryption schemes were developed for image encryption to achieve low computational complexity [19, 20]. However, these algorithms have a low level of security due to their predictable decomposition results and/or limited key spaces [16]. Our previous work proposed image encryption algorithms using the parametric decomposition methods including the (n, k, p)-Gray code decomposition [14] and Fibonacci p-code decomposition [16]. However, there still exists a room for security enhancement.

To further improve security of the bitplane decomposition based image encryption methods, this paper introduces a new image encryption algorithm using a bitplane of a source image. The algorithm selects a bitplane decomposition method to generate a bitplane of a source image. This bitplane acts as the security key bitplane to encrypt bitplanes of the original image using the XOR operations.

* Corresponding author. Tel.: +853 83978458; fax: +853 28838314.
E-mail address: yicongzhou@umac.mo (Y. Zhou).

0165-1684/$ - see front matter © 2014 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.sigpro.2014.01.020

The proposed algorithm then applies a scrambling algorithm for bit-level permutation and combines all processed bitplanes to obtain the encrypted image. It can use any image as the source image, any bitplane decomposition method to generate the security key bitplane, and any scrambling algorithm for bit-level permutation. Simulation results and security analysis are provided.

This paper is organized as follows: Section 2 reviews three bitplane decomposition methods which will be used as examples of existing bitplane decomposition approaches for the new image encryption algorithm introduced in Section 3. Simulation results and security analysis are presented in Sections 4 and 5, respectively. Section 6 reaches a conclusion.

2. Bitplane decomposition methods

This section briefly reviews three bitplane decomposition technologies including the binary bitplane decomposition (BBD), Gray code bitplane decomposition (GCBD), and truncated Fibonacci p-code bitplane decomposition (TFPBD). BBD and GCBD are two traditional decomposition methods. They can decompose a grayscale image[1] into 8 binary bitplanes. TFPBD is a parametric bitplane decomposition method. These methods will be applied to generate the security key bitplane for the new image encryption algorithm proposed in Section 3.

2.1. Binary bitplane decomposition

A non-negative decimal number N can be represented by a binary sequence $(b_{n-1},\ldots,b_1,b_0)$ based on the following equation:

$$N=\sum_{i=0}^{n-1} b_i 2^i = b_0 2^0 + b_1 2^1 + \cdots + b_{n-1}2^{n-1} \tag{1}$$

Because pixel values in a grayscale image are decimal numbers between 0 and 255, each pixel can be represented by an 8-bit binary sequence. Thus, BBD can decompose a grayscale image into 8 binary bitplanes (BBs) [14,16,21]. The ith bitplane consists of all the ith bits of the binary representation of each pixel within the grayscale image. Among these bitplanes, higher bitplanes contain more significantly visual information of the original image while lower bitplanes show more details. Fig. 1(a) shows the 6th BB of the grayscale image in Fig. 5(e).

2.2. Gray code bitplane decomposition

Gray code is known as the binary reflected Gray code in which two successive codes change only one bit position. It can be defined in Eq. (2) where $(b_{n-1},\ldots,b_1,b_0)$ and $(g_{n-1},\ldots,g_1,g_0)$ are the binary representation and Gray code of the nonnegative decimal number N, respectively.

$$g_i=\begin{cases} b_i, & i=n-1 \\ b_i \oplus b_{i+1}, & 0\le i\le n-2 \end{cases} \tag{2}$$

[1] A grayscale image in this paper denotes an image with pixel values within [0, 255].

Similar to BBD, a grayscale image can also be decomposed into 8 Gray code bitplanes (GCBs). However, for a specific image, the contents of each bitplane in GCBD are totally different from these of the corresponding bitplane in BBD. The advantage of this decomposition method is that it can reduce the effect of small gray-level changes [21]. A GCBD example is shown in Fig. 1(b).

2.3. Truncated Fibonacci p-code bitplanes decomposition

Motivated by the concept of binary representation, a non-negative decimal number N can be represented by the truncated Fibonacci p-code $(t_{n-1},\ldots,t_2,t_1,t_0)_p$ using

$$N=\sum_{i=0}^{n-1} t_i T_p(i) = t_0 T_p(0)+t_1 T_p(1)+\cdots+t_{n-1}T_p(n-1) \tag{3}$$

where t_i is the weight coefficient and $t_i\in[0,1]$, $T_p(i)$ is the truncated Fibonacci p-code sequence defined by

$$T_p(i)=\begin{cases} 0, & i<0 \\ 1, & i=0 \\ F_p(i+p), & i>0 \end{cases} \tag{4}$$

where i is the index number; the non-negative integer p is a distance parameter; and $F_p(i+p)$ is the p-Fibonacci sequence defined by

$$F_p(i)=\begin{cases} 0, & i<0 \\ 1, & i=0 \\ F_p(i-1)+F_p(i-p-1), & i>0 \end{cases} \tag{5}$$

Using the constrains in [16], a decimal number can be uniquely represented by a truncated Fibonacci p-code $(t_{n-1},\ldots,t_2,t_1,t_0)_p$ with a certain p value.

Similar to BBD, an image can also be decomposed into the truncated Fibonacci p-code bitplanes (TFPBs). Different from BBD and GCBD, the truncated Fibonacci p-code bitplanes decomposition (TFPBD) is a parameter-dependent decomposition method. For a specific image, TFPBD can generate more than eight bitplanes. For instance, a grayscale image can be decomposed into twelve bitplanes with $p=1$. Moreover, the number of bitplanes and contents in each bitplane are different with a change of p values. For example, when $p=2$, the previous gray image can be decomposed into 14 bitplanes. Fig. 1(c)–(d) show the 6th TFPB of the Lena image with different p values.

In summary, for a given image, those decomposition methods can generate different numbers of bitplanes and contents in each bitplane also differ. As can be seen in Fig. 1, bitplanes with the same level in these bitplane decomposition methods are totally different. These methods will be used for generating the security key bitplane in the new image encryption algorithm proposed in Section 3.

3. New image encryption algorithm

In this section, we propose a novel image encryption algorithm, named DecomCrypt, based on three bitplane decomposition methods presented in Section 2. The algorithm can effectively encrypt the grayscale, color, biometric, and medical images.

Fig. 1. Image bitplane decomposition of the grayscale Lena image using different methods. (a)–(d) show the 6th bitplane decomposed by the (a) BBD, (b) GCBD, (c) TFPBD with $p=1$, and (d) TFPBD with $p=2$.

The block diagram of DecomCrypt is shown in Fig. 2. The algorithm first decomposes the original image (the image to be encrypted) into eight binary bitplanes using BBD. To change bit values, each bitplane is then performed the XOR operation with a security key bitplane, individually. A scrambling algorithm is used to change all bit locations. After n iterations of the XOR and scrambling processes, the DecomCrypt algorithm combines all bitplanes to obtain the encrypted image.

As can be seen from Fig. 2, the security key bitplane is selected from the bitplanes of a source image generated by one of the decomposition methods presented in Section 2. However, other bitplane decomposition methods can also be used here. The source image could be any existing or newly generated image with the same size of the original image.

Algorithm 1. The scrambling algorithm.

Input: Input bitplanes $B(m,n,k)$ with a size of $L=M\times N\times 8$, and parameters a and X_0

1: Reshape $B(m,n,k)$ into a 1D binary matrix $E=\{E_1,E_2,\ldots,E_L\}$, where $E_{(m-1)N+n+(k-1)L/8}=B(m,n,k)$

2: $c\leftarrow$ number of 1's in E, $\tilde{c}=c \bmod 512$

3: Iterate the Logistic map $X_{p+1}=aX_p(1-X_p)$ for $\tilde{c}$ times and update the initial value $X_0=X_{\tilde{c}}$

4: Re-iterate the Logistic map for L times and generate a 1D matrix $X=\{X_1,X_2,\ldots,X_L\}$

5: Sort X with data values from low to high and obtain a matrix $X'=\{X'_1,X'_2,\ldots,X'_L\}$, which is a permutation of X

6: Generate another 1D data matrix $E'=\{E'_1,E'_2,\ldots,E'_L\}$ where $E'_j=E_i$ for $X'_j=X_i$. Therefore, E' is a permutation of E

7: Reshape matrix E' to $M\times N\times 8$ bitplanes as $B'(m,n,k)=E'_{(m-1)N+n+(k-1)L/8}$

Output: Scrambled bitplanes $B'(m,n,k)$

Any existing or new scrambling method can be used in DecomCrypt. As an example, this paper introduces a new bit-level scrambling algorithm to change bit positions within each XORed bitplane or/and across all bitplanes. It is described in Algorithm 1. As we can see from this scrambling algorithm, parameter $\tilde{c}$ is designed to record the content changes of the input bitplanes and update the initial value of the Logistic map. For different input bitplanes, the matrix X significantly changes, resulting in completely different scrambled bitplanes B'. This ensures that DecomCrypt withstands the differential attack.

The security keys of DecomCrypt consist of the iterations n, source image or its location (the location of an image database or a link of webpage), decomposition method and its parameters, location of the security key bitplane, as well as the scrambling algorithm and its security keys. These security keys are encoded as messages or emails and then transmitted over separated security channels. This ensures that they are safely and correctly delivered to the authorized users for image decryption. The users can follow the inverse processes of image encryption in Fig. 2 to recover the original image.

In summary, the proposed DecomCrypt has at least five following advantages.

(1) Any new or existing image with the same size of the original image can be used as the source image.
(2) Any bitplane decomposition method can be used to generate the security key bitplane.
(3) The users have the flexibility to utilize a various number of security key bitplanes for image encryption. This ensures the attacker's difficulty of finding the correct security key bitplane.
(4) Any scrambling algorithm can be used to change bit positions.
(5) DecomCrypt has a significantly large key space against the Brute-force attack.

4. Simulation results

The proposed DecomCrypt is able to encrypt different types of images such as grayscale, color, biometric and medical images. This section provides simulation results to show its encryption performance.

Fig. 3 shows encryption examples of grayscale, color, biometric and medical images. For color image encryption, we encrypt each color component individually and combine them to obtain the encrypted color images. As can be seen, all encrypted images (Fig. 3(d)) are noise-like and unrecognizable when choosing different source images (Fig. 3(b)), decomposition methods, or bitplanes (Fig. 3(c)). DecomCrypt can well protect different types of original images (Fig. 3(a)). The reconstructed images in Fig. 3(e) are visually the same as their original ones in Fig. 3(a). This means that the original information can be fully recovered in the decryption process.

The iterative encryption processes have great effects on the encryption performance of DecomCrypt. Fig. 4 shows

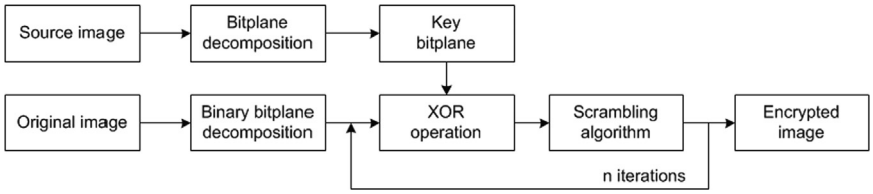

Fig. 2. The DecomCrypt algorithm.

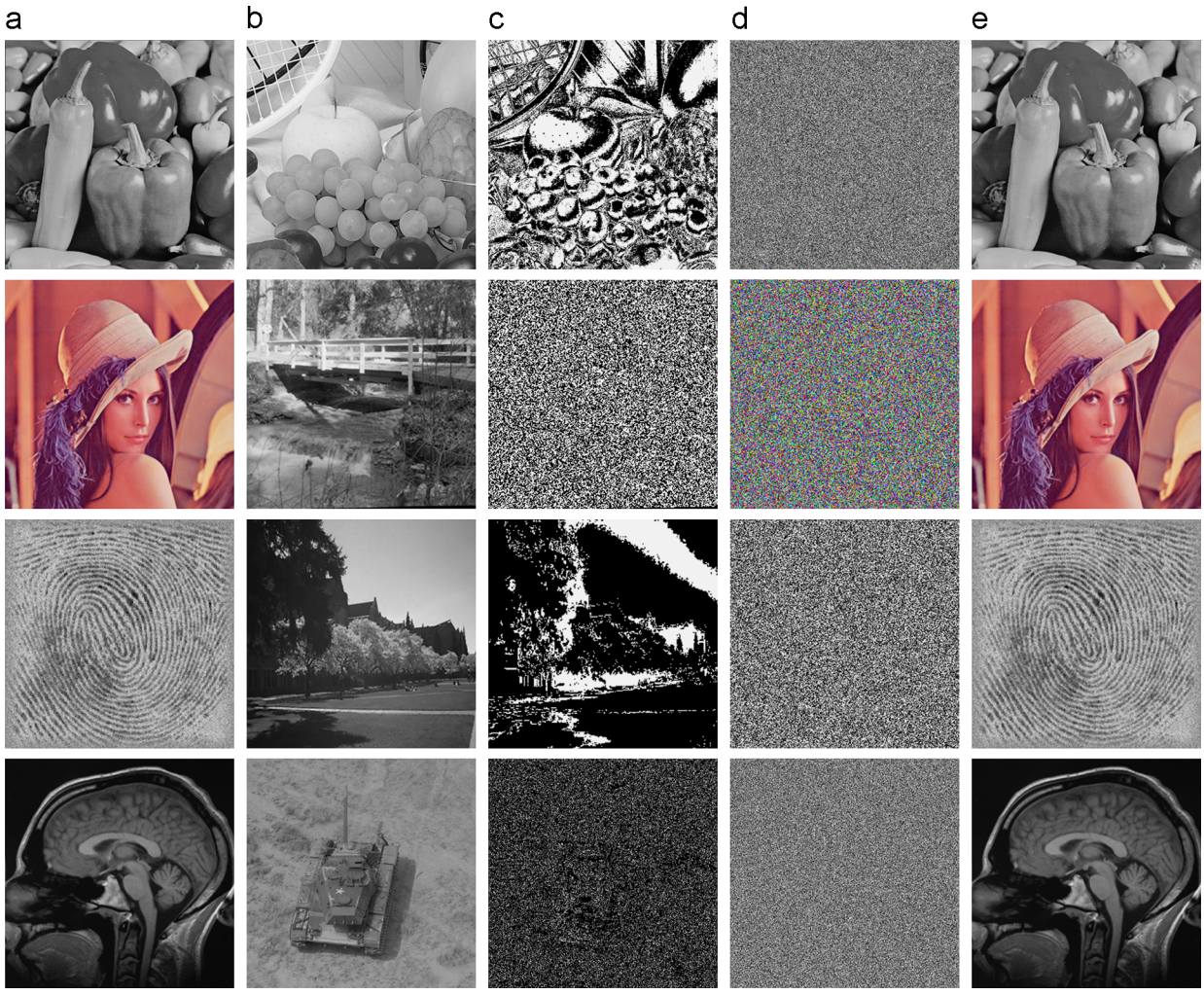

Fig. 3. Image encryption using different bitplane decomposition methods. The 1st–4th rows show the encryption results of the grayscale, color, biometric and medical images, respectively. (a) The original images; (b) the source images; (c) the security key bitplanes: from top to bottom, the 5th GCB, 1st BB, 9th TPFB with $p=2$, and 4th TPFB with $p=3$, respectively; (d) the encrypted images; (e) the recovered images.

the encryption results of DecomCrypt with different iterations. As can be seen, with the encryption iteration increasing, histograms of the encrypted images become flat or uniformly distributed, indicating better encryption performance. On the other hand, in image decryption, the original image can be completely reconstructed only in the correct iteration such as the image in Fig. 4(h). Otherwise, even using the correct security keys, the reconstructed images in incorrect iterations are unrecognizable such as images in Fig. 4(f) and (g).

5. Performance and security analysis

This section addresses the performance and security of the proposed DecomCrypt. We analyze its key space and key sensitivity, and then evaluate DecomCrypt using histogram

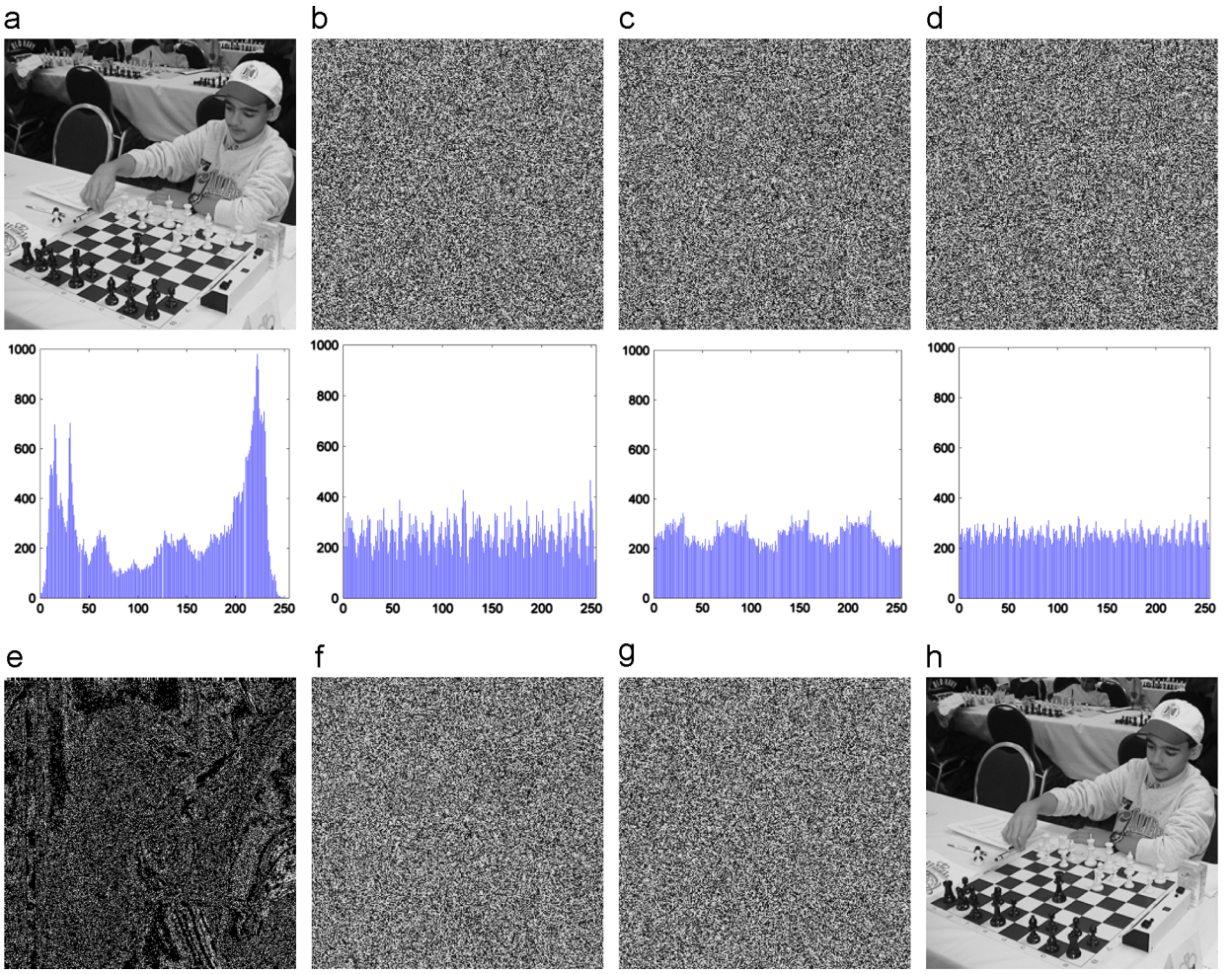

Fig. 4. Image encryption with different iterations. (a) The original image and its histogram; (b)–(d) show the encrypted images and their histograms after the (b) first, (c) second, and (d) third iterations, respectively; (e) the 4th TPFB with $p=2$, (f)–(h) are images recovered from (d) in the (f) first, (g) second, and (h) third iterations, respectively.

analysis, correlation analysis, encryption efficiency analysis and the differential attack.

5.1. Security key space

The key space denotes the total number of possible combinations of the algorithm's security keys. The Brute-force attack is one common attack in which an attacker attempts to guess the correct security keys by excessively searching the key space of an encryption algorithm. Thus, a sufficient large key space can ensure that the encryption algorithm withstands the Brute-force attack. As mentioned in Section 3, the security keys of DecomCrypt consist of four portions: (1) the source image or its location, (2) the bitplane decomposition method and its parameter if any, (3) the location of the security key bitplane within the decomposed ones, (4) the scrambling algorithm and its security keys.

To calculate the key space of DecomCrypt, as an example, we choose a grayscale image with a size of $M \times N$ as the original image. Assume the source image can be selected from m images with the same size of the original one, then the possible number of the source image is $S_1 = m$.

Suppose we use only three bitplane decomposition methods (BBD, GCBD, and TFPBD) to generate the security key bitplane. The number of possible security key bitplanes using BBD or GCBD is 8. For TFPBD, the number of security key bitplanes changes with p values. Suppose there are n_k bitplanes when p value is k. The possible choices of the security key bitplane using BBD, GCBD, and TFPBD will be $S_2 = 8+8+\sum_{p=0}^{k} n_p$.

The original image is decomposed by BBD into 8 bitplanes. Any existing or new scrambling algorithm can be used for changing bit positions. Their possible changes in this process are $S_3 = M! \times N! \times 8!$

Therefore the key space of DecomCrypt is $S = S_1 \times S_2 \times S_3 = m \times (16+\sum_{p=0}^{k} n_p) \times M!N!8!$. For example, suppose we want to encrypt a 100×100 grayscale image, the source image has 10 possible choices, and choose the TFPBD with p value 2 to generate the security key bitplane (14 possible bitplanes). The key space of DecomCrypt is $S = 10 \times 14 \times 100! \times 100! \times 8! = 4.916 \times 10^{322}$. It is large enough to resist the Brute-force attack.

5.2. Key sensitive test

A good image encryption algorithm should have high sensitivity to its security key changes. Here, we test the key sensitivity of the DecomCrypt in both the encryption and decryption processes. For simplicity unless specified, the rest of this paper uses the Lena in Fig. 5(e) as the original image and the Building image in Fig. 5(a) as the source image to show our simulation results in detail.

As the example of image encryption shown in Fig. 5, we use the 3rd TFPB (Fig. 5(b)) of the source image (Fig. 5(a)) as the security key bitplane to encrypt the original image (Fig. 5(e)) and obtain the encrypted image shown in Fig. 5(f). In the similar way, we choose the 6th TFPB (Fig. 5(c)) of the source image as another security key bitplane to encrypt the same original image in Fig. 5(a). The encrypted image is shown in Fig. 5(g). From the differences of these two encrypted images shown in Fig. 5(h), the encrypted image in Fig. 5(g) is significantly different from the encrypted one in Fig. 5(f). This demonstrates that a slight change in the security key of DecomCrypt (i.e. the bitplane position is changed from the 3rd to 6th.) results in a new security key bitplane and thus a completely different encrypted image.

For image decryption shown in Fig. 6, we use the 2nd BB of the source image in Fig. 5(a) as the security key bitplane to encrypt the original image. The encrypted image is shown in Fig. 6(c). We then use different security key bitplanes to recover the original image. Fig. 6(d) shows the decryption result using the incorrect security key bitplane 2nd GCB (Fig. 6(b)) of the source image. As can be seen, the decrypted image is noise-like without leaking any original information. Fig. 6(e) shows the decrypted image using the correct security key bitplane in Fig. 6(a). This shows that the original image can be completely recovered only the correct security key is being utilized. Therefore, the proposed DecomCrypt is also sensitive to its key changes during the decryption process.

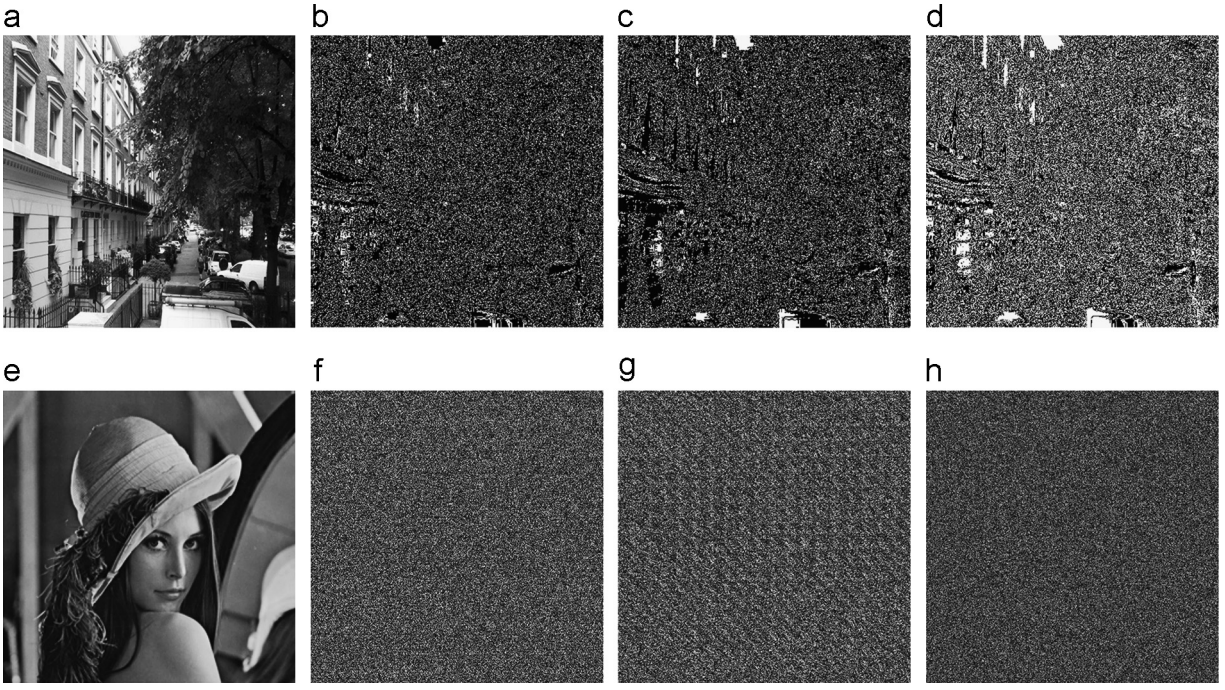

Fig. 5. Key sensitivity test for image encryption. (a) The source image; (b) the 3rd TFPB, $p=2$; (c) the 6th TPFB, $p=2$; (d) the difference between (b) and (c); (e) the original image; (f) the encrypted image using (b); (g) the encrypted image using (c); (h) the difference between (f) and (g).

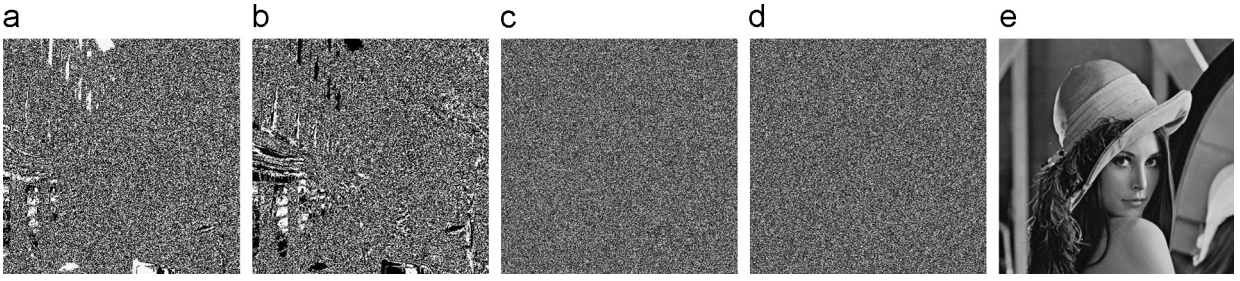

Fig. 6. Key sensitivity test for image decryption. (a) The security key bitplane 2nd BB, and (b) the 2nd GCB of the source image in Fig. 5(a); (c) the encrypted image using (a) as the security key bitplane; (d) and (e) are recovered images using incorrect security key bitplane (b) and correct security key bitplane (a), respectively.

5.3. Histogram analysis

Image histogram represents the intensity distribution of pixels within an image. An encrypted image with a flat histogram is able to resist statistic attacks. To show the encryption performance of DecomCrypt, we compare the histogram of its encrypted image with those by Zhu's algorithm [8] and Zhou's algorithm [14]. The results are shown in Fig. 7.

The original Lena image (Fig. 7) is encrypted by three algorithms. The encrypted images and their histograms are shown in Fig. 7(b)–(d). As can be seen, the encrypted images by the DecomCrypt and Zhu's algorithm are noise-like images with uniform distributions. The encrypted image by Zhou's algorithm in Fig. 7(c) has a nonuniform distribution. This means that DecomCrypt has excellent encryption performance which is similar to Zhu's algorithm but better than Zhou's algorithm. These features ensure that the images encrypted by DecomCrypt are able to withstand statistic attacks.

5.4. Correlation analysis

An encryption algorithm intends to break the relationship of adjacent pixels within an image and prevent the leakage of the original information. Correlation analysis here is to test the relationships of adjacent pixels in the original and encrypted images. 3000 pairs of two adjacent pixels are randomly chosen from the original and encrypted images in the horizontal, vertical, and diagonal directions to perform this analysis.

We then set the intensity values of adjacent pixel pairs as the horizontal and vertical axes, respectively. Fig. 8 plots their distributions in three directions. If two adjacent pixels are equal, their tracks are located in the diagonal line. As can be seen, the distributions of adjacent pixels in the original image are around the diagonal line. This means the original image has high correlation. On the contrary, the distributions of adjacent pixels in the encrypted image disperse in the whole data range of the image. This proves that the encrypted image has extremely low correlation and shows high randomness.

To quantitatively perform correlation analysis, we use Eq. (6) to calculate correlation coefficients of adjacent pixels at the horizontal, vertical and diagonal directions [22].

$$corr(x,y) = \frac{E[(x-\mu_x)(y-\mu_y)]}{\sigma_x \sigma_y} \tag{6}$$

where μ_x and μ_y are the mean values of x and y respectively, and σ_x and σ_y denote the standard deviations of x and y respectively; $E[\cdot]$ is the expectation function.

A correlation coefficient approaching to 1 indicates a strong correlation while the coefficient close to 0 means extremely low correlation. We use Eq. (6) to quantitatively evaluate the correlations of adjacent pixels in the original and encrypted images in Fig. 8(a). The results are shown in Table 1.

As shown in Table 1, at all three directions, the correlation coefficients of the original image are close to 1 while those of the encrypted image are around zero. This further verifies that the encrypted image has good randomness and its adjacent pixels have extremely low correlation.

5.5. Encryption efficiency analysis

In this section, we analyze the encryption speed of DecomCrypt with respect to different iterations and image sizes. The original image with 256×256 and the key bitplane are the ones shown in Fig. 4(a) and (e).

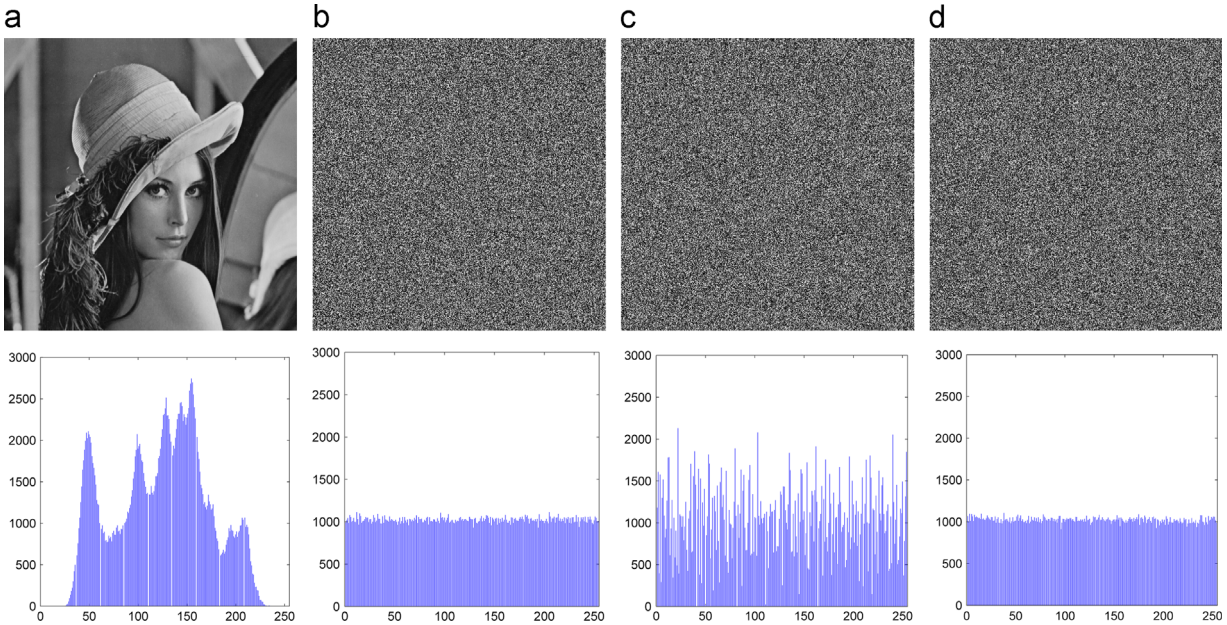

Fig. 7. Histogram analysis. (a) The original image and its histogram; (b)–(d) are the encrypted images and their histograms using (b) Zhu's algorithm [8], (c) Zhou's algorithm [14], and (d) the DecomCrypt.

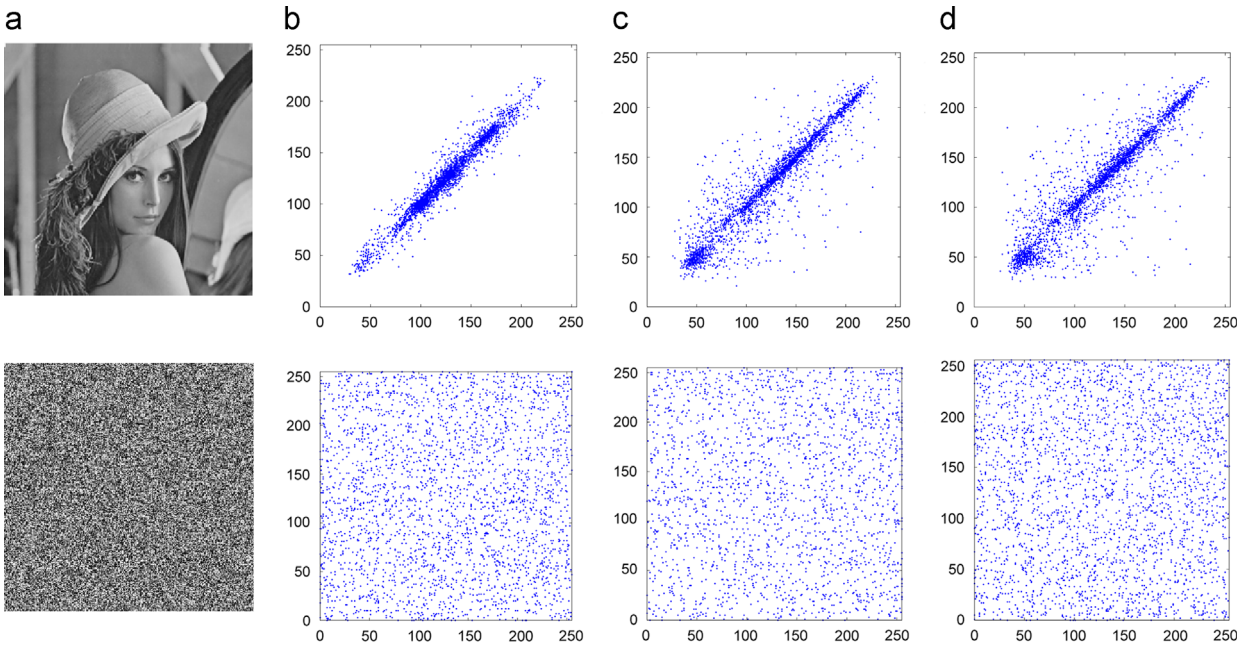

Fig. 8. Correlation of adjacent pixels at different directions. The top and bottom rows show adjacent pixel correlation of (a) the original and encrypted images at the (b) horizontal, (c) vertical, (d) diagonal directions, respectively.

Table 1
Correlation coefficients of the original and encrypted images.

Image	Horizontal	Vertical	Diagonal
Original image in Fig. 8(a)	0.9750	0.9628	0.9439
Encrypted image in Fig. 8(a)	0.0102	−0.0053	−0.0161

Experiments are carried out in Matlab 2010a in a computer with an Intel Core i7 2.8 GHz and 4 GB RAM running the Windows 7 operating system. Fig. 9 plots the encryption time changing with the iterations and the original image sizes.

Fig. 9(a) plots the encryption time with respect to the encryption iterations. When the iteration increases from 1 to 20, the encryption time of DecomCrypt increases only 1.676 s. This means the iteration has limited effect on the encryption speed of DecomCrypt. We also compare the encryption speed of DecomCrypt with Zhu's [8] and Zhou's [14] algorithms when the size of the original image increases from 16×16 to 256×256 with a step of 16. The results are plotted in Fig. 9(b). As can be seen, DecomCrypt has the fastest encryption speed compared with other two algorithms. Moreover, experiments in Fig. 4 have proved that three iterations are sufficient for DecomCrypt to achieve excellent encryption performance. Thus, three iterations are recommended to balance the tradeoff between the encryption performance and the speed requirement in real-time applications.

5.6. Differential attack

The differential attack is a chosen-plaintext attack in which the attacker intends to recover the security key by slightly changing the original images and then exploring the changes of the corresponding encrypted images. To withstand the differential attack, a well-designed encryption algorithm should ensure that a tiny change in the original image will result in a significant change in the encrypted image.

To evaluate the performance against the differential attack, the proposed DecomCrypt is compared with Zhu's algorithm [8] and Zhou's algorithm [14]. The results are shown in Fig. 10.

In experiments shown in Fig. 10, two original images with only one-pixel difference are encrypted by three encryption algorithms, Zhu's algorithm [8], Zhou's algorithm [14] and DecomCrypt. As can be seen, a tiny change in the original image results in a significant change only in the encrypted image by DecomCrypt. However, this tiny change has almost no effect on the encrypted images by Zhu's and Zhou's algorithms. These can be verified by image differences shown in the bottom rows in Fig. 10(b) and (c). DecomCrypt performs better than other two encryption algorithms.

Two measures are used to quantitatively evaluate this impact of a tiny change in the original image to the encrypted images. They are the number of pixel change rate (NPCR) and unified average changing intensity (UACI) defined by

$$UACI = \frac{1}{MN}\sum_{i=1}^{M}\sum_{i=1}^{N}\left(\frac{|E_1(i,j)-E_2(i,j)|}{255}\right)\times 100\% \tag{7}$$

$$NPCR = \frac{\sum_{i=1}^{M}\sum_{j=1}^{N}\mathcal{A}(i,j)}{MN}\times 100\% \tag{8}$$

where E_1 and E_2 are two encrypted images with a size of $M\times N$, $\mathcal{A}(i,j)$ is the number of different pixels between E_1 and E_2.

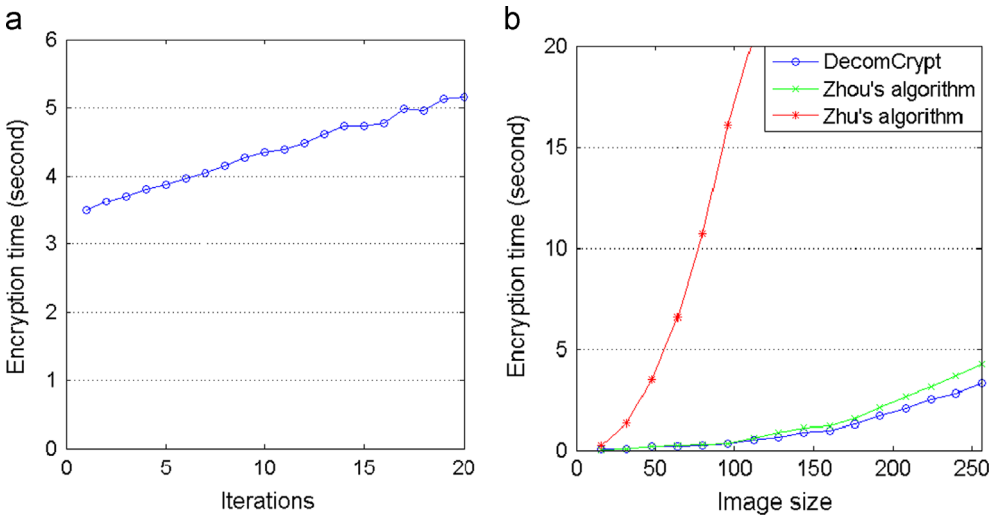

Fig. 9. Encryption efficiency analysis. (a) Encryption time with different iterations; (b) encryption time comparison of different algorithms when the original image size increases.

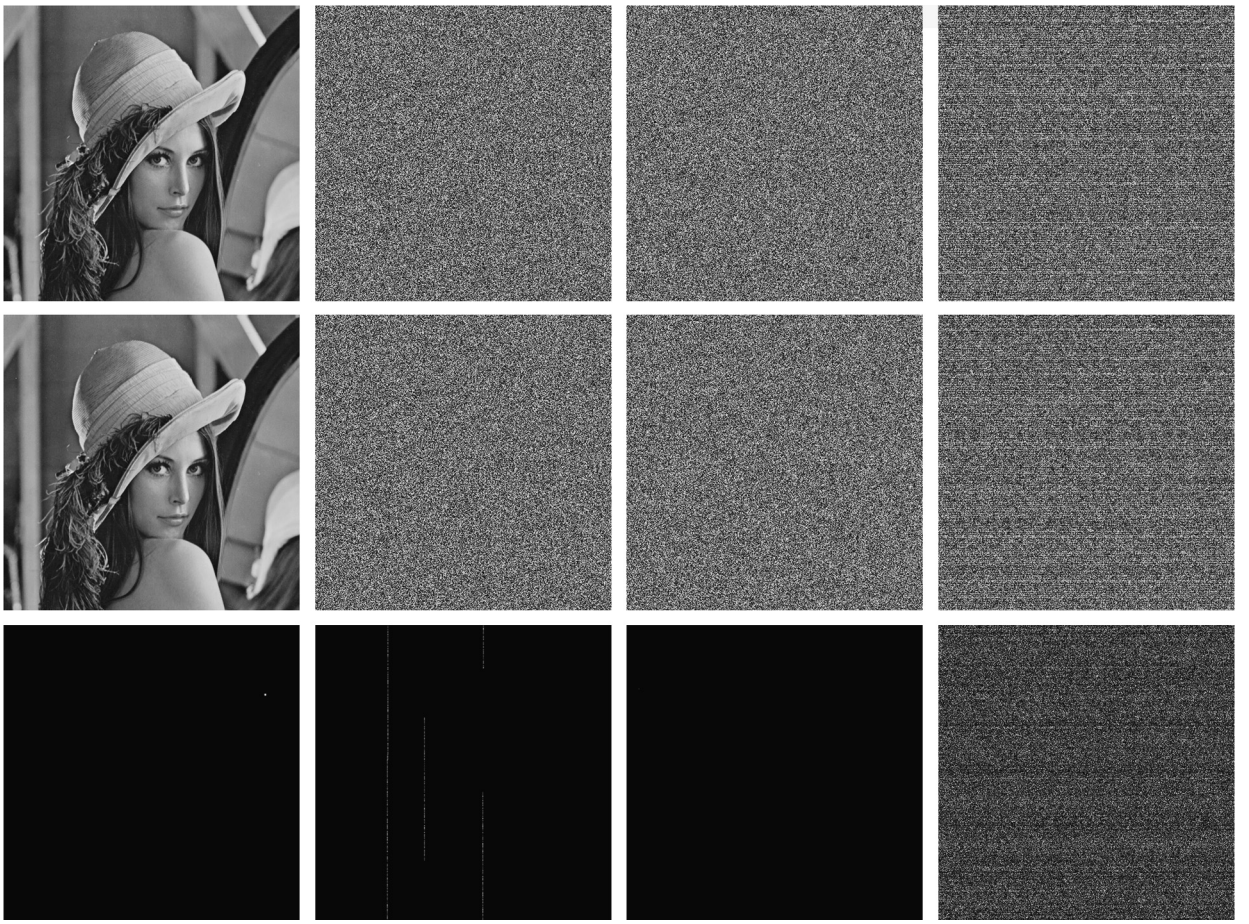

Fig. 10. Differential attack to different algorithms. (a) The original images and their differences, (b)–(d) are the encrypted images and their differences using (b) Zhu's algorithm [8], (c) Zhou's algorithm [14] and (d) the DecomCrypt.

Table 2
Comparison of different encryption algorithms against the differential attack.

Image	NPCR			UACI		
	Zhu's	Zhou's	DecomCrypt	Zhu's	Zhou's	DecomCrypt
1	0.0048	0.000003815	0.9959	0.0016	0.000000314	0.3454
2	0.0012	0.000003815	0.9959	0.0004	0.000002244	0.3342
3	0.0024	0.000003815	0.9962	0.0008	0.000000254	0.3363
4	0.0032	0.000003815	0.9376	0.0011	0.000004473	0.2879
5	0.0027	0.000003815	0.9964	0.0009	0.000002109	0.3703
6	0.0033	0.000003815	0.9963	0.0011	0.000000449	0.3445
7	0.0004	0.000003815	0.9962	0.0001	0.000001586	0.3341
8	0.0063	0.000003815	0.9959	0.0021	0.000000180	0.3342
9	0.0050	0.000003815	0.9960	0.0018	0.000000554	0.3344
10	0.0024	0.000003815	0.9961	0.0008	0.000000464	0.3346
11	0.0014	0.000003815	0.9380	0.0005	0.000002334	0.2871
12	0.0059	0.000003815	0.7531	0.0020	0.000000793	0.2661
13	0.0014	0.000003815	0.9961	0.0005	0.000000359	0.3446
14	0.0053	0.000003815	0.9374	0.0018	0.000002872	0.2864
15	0.0010	0.000003815	0.9963	0.0004	0.000005595	0.3456
16	0.0019	0.000003815	0.9960	0.0006	0.000004907	0.3350
17	0.0012	0.000003815	0.9963	0.0004	0.000005341	0.3347
18	0.0002	0.000003815	0.7489	0.0007	0.000006118	0.2589
19	0.0050	0.000003815	0.9961	0.0017	0.000000374	0.3461
20	0.0000	0.000003815	0.9961	0.0000	0.000003471	0.3699
21	0.0099	0.000003815	0.9962	0.0034	0.000004563	0.3461
22	0.0009	0.000003815	0.9959	0.0003	0.000001331	0.3765
23	0.0013	0.000003815	0.9377	0.0005	0.000000927	0.3319
24	0.0037	0.000003815	0.9959	0.0012	0.000000105	0.3691
25	0.0070	0.000003815	0.9963	0.0023	0.000005984	0.3348
26	0.0000	0.000003815	0.9374	0.0000	0.000001316	0.2921
27	0.0031	0.000003815	0.9963	0.0010	0.000000643	0.3706
28	0.0030	0.000003815	0.9961	0.0011	0.000000613	0.3446
29	0.0085	0.000003815	0.7506	0.0029	0.000005984	0.2592
30	0.0016	0.000003815	0.9377	0.0005	0.000003605	0.2870
31	0.0002	0.000003815	0.9962	0.0001	0.000002708	0.4725
32	0.0000	0.000003815	0.9388	0.0000	0.000002887	0.3145
33	0.0039	0.000003815	0.9959	0.0014	0.000006373	0.3465
33b	0.0044	0.000003815	0.9374	0.0015	0.000001975	0.2912
34	0.0024	0.000003815	0.9960	0.0008	0.000003560	0.3341
35	0.0042	0.000003815	0.9960	0.0014	0.000003560	0.3358
36	0.0013	0.000003815	0.9383	0.0004	0.000003605	0.2884
37	0.0008	0.000003815	0.9961	0.0003	0.000006328	0.3470
38	0.0014	0.000003815	0.9957	0.0005	0.000006463	0.3348
39	0.0008	0.000003815	0.9958	0.0003	0.000007061	0.3340
Average	0.0029	0.000003815	0.9646	0.0009	0.000002859	0.3310

We select first 40 test images from the website.[2] These images are grayscale images with a size of 512×512. They are applied with the same differential attack as those in Fig. 10. We then use NPCR and UACI to measure the encrypted images by three encryption algorithms. The measure results are shown in Table 2. As can be seen, DecomCrypt obtains NPCR 96.46% and UACI 33.10% in average. However, the average NPCR and UACI values of Zhu's algorithm are 0.29% and 0.09%; and of Zhou's algorithm are 0.0003815% and 0.0002859%, respectively. This further demonstrates that DecomCrypt outperforms these two algorithms in terms of the differential attack.

[2] http://decsai.ugr.es/cvg/dbimagenes/g512.php

6. Conclusion

This paper proposed a novel image encryption algorithm integrating the bit-level permutation with three bitplane decomposition technologies: binary bitplane decomposition, Gray code bitplane decomposition and truncated Fibonacci p-code bitplane decomposition. The proposed algorithm uses a bitplane of a source image as the security key bitplane for protecting image contents. It has a huge key space because any image could act as the source image and any image scrambling method could be used for changing bit positions. Simulations and security analysis have demonstrated that the proposed algorithm is highly sensitive to its security key, shows excellent encryption performance for protecting different types of images, and outperforms state-of-art algorithms in terms of security and encryption performance. It has potential applications for multimedia communications.

Acknowledgment

This work was supported in part by the Macau Science and Technology Development Fund under Grant 017/2012/A1 and by the Research Committee at University of Macau under Grants MYRG113(Y1-L3)-FST12-ZYC and MRG001/ZYC/2013/FST.

References

[1] H. Cheng, X. Li, Partial encryption of compressed images and videos, IEEE Trans. Signal Process. 48 (8) (2000) 2439–2451.
[2] Y. Zhou, K. Panetta, R. Cherukuri, S. Agaian, Selective object encryption for privacy protection, in: SPIE Defense, Security, and Sensing 2009: Mobile Multimedia/Image Processing, Security, and Applications 2009, vol. 7351, Orlando, Florida, 2009, pp. 73510F–10.
[3] N. Zhou, T. Dong, J. Wu, Novel image encryption algorithm based on multiple-parameter discrete fractional random transform, Opt. Commun. 283 (15) (2010) 3037–3042.
[4] National Institute of Standards and Technology, Data encryption standard (DES), 1999. URL ⟨http://csrc.nist.gov/publications/fips/fips46-3/fips46-3.pdf⟩.
[5] J. Daemen, V. Rijmen, AES proposal: Rijndael, 1999. URL ⟨http://csrc.nist.gov/archive/aes/rijndael/Rijndael-ammended.pdf⟩.
[6] Y. Zhou, L. Bao, C.L.P. Chen, Image encryption using a new parametric switching chaotic system, Signal Process. 93 (11) (2013) 3039–3052.
[7] X. Wang, L. Teng, X. Qin, A novel colour image encryption algorithm based on chaos, Signal Process. 92 (2012) 1101–1108.
[8] Z.-L. Zhu, W. Zhang, K.-W. Wong, H. Yu, A chaos-based symmetric image encryption scheme using a bit-level permutation, Inf. Sci. 181 (6) (2011) 1171–1186.
[9] T.H. Chen, K.H. Tsao, Y.S. Lee, Yet another multiple-image encryption by rotating random grids, Signal Process. 92 (9) (2012) 2229–2237.
[10] X. Liao, S. Lai, Q. Zhou, A novel image encryption algorithm based on self-adaptive wave transmission, Signal Process. 90 (2010) 2714–2722.
[11] L. Chen, D. Zhao, Color information processing (coding and synthesis) with fractional fourier transforms and digital holography, Opt. Express 15 (24) (2007) 16080–16089.
[12] F. Mosso, M. Tebaldi, J.F. Barrera, N. Bolognini, R. Torroba, Pure optical dynamical color encryption, Opt. Express (2011) 13779–13786.
[13] F. Auli-Llinas, M.W. Marcellin, Scanning order strategies for bitplane image coding, IEEE Trans. Image Process. 21 (4) (2012) 1920–1933.
[14] Y. Zhou, K. Panetta, S. Agaian, C.L.P. Chen, (n, k, p)-Gray code for image systems, IEEE Trans. Cybern. 43 (2) (2013) 515–529.
[15] S. Agaian, J. Astola, K. Egiazarian, P. Kuosmanen, Decompositional methods for stack filtering using Fibonacci p-codes, Signal Process. 41 (1995) 101–110.
[16] Y. Zhou, K. Panetta, S. Agaian, C.L.P. Chen, Image encryption using p-Fibonacci transform and decomposition, Opt. Commun. 285 (5) (2012) 594–608.
[17] D.Z. Gevorkian, K.O. Egiazarian, S.S. Agaian, J.T. Astola, O. Vainio, Parallel algorithms and VLSI architectures for stack filtering using Fibonacci p-codes, IEEE Trans. Signal Process. 43 (1) (1995) 286–295.
[18] J.-W. Han, C.-S. Park, D.-H. Ryu, E.-S. Kim, Optical image encryption based on XOR operations, Opt. Eng. 38 (1) (1999) 47–54.
[19] M. Podesser, H. Schmidt, A. Uhl, Selective bitplane encryption for secure transmission of image data in mobile environments, in: The 5th Nordic Signal Processing Symposium—NORSIG-2002, 2002, p. 1037.
[20] D. Moon, Y. Chung, S.B. Pan, K. Moon, K. Chung, An efficient selective encryption of fingerprint images for embedded processors, Electron. Telecommun. Res. Inst. J. 28 (4) (2006) 444–452.
[21] R.C. Gonzalez, R.E. Woods, Digital Image Processing, 3rd edition, Pearson Prentice Hall, Upper Saddle River, New Jersey, 2008 ISBN 9780131687288.
[22] A.G. Bluman, Elementary Statistics: A Step by Step Approach, McGraw-Hill, Boston, 1997.