A Block-based Image Encryption Algorithm in Frequency Domain using Chaotic Permutation

Rinaldi Munir
School of Electrical Engineeringline and Informatics
Institute Technology of Bandung, ITB
Bandung, West Java, Indonesia
rinaldi-m@stei.itb.ac.id

***Abstract*—Images can be encrypted in spatial domain, frequency domain, or both. Most of the image encryption algorithms operate in spatial domain. Encryption in frequency domain has an advantage of being resistante to many image processing operations. This paper presents an image encryption algorithm in frequency domain using a chaotic permutation. Because of the images represented in JPEG format, then to increase robustness to the image processing operations, the plain images are encrypted in blocks 8 x 8. After an image is encrypted, the cipher-image was modified by some common image processings. The experiments shows that the cipher-image was robust to the image processing operation, the decrypted images can be still recognized well.**

Keywords*: *image encryption, frequency domain, robustness.

I. INTRODUCTION

Images are kind of information that play important role in current information technology era. Images can be stored in the storage devices or transmitted through the communication channel. However, store or transmit them in the plain form have risks. The adversary can steal or tap the plain-images, so that confidentiality of plain-images need to be protected. Images encryption is the solution to this problem. The plain images are encrypted so that it can not be recognized by unauthorized party (see Fig. 1).

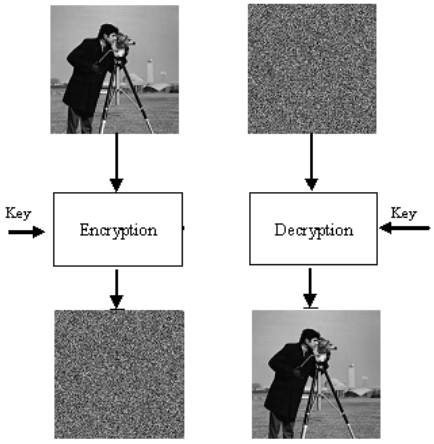


Fig.1. Encryption and decryption of digital images

Many image encryption algorithms have been proposed. Most of the algorithms operate in spatial domain, usually by modifying the most significant bit(s) (or MSB) of the cipher images using an encryption key. Encryption in spatial domain has disadvantage, namely once the cipher image is modified (by common image processing operations such as JPEG compression, adding noise, brightness/contrast adjustment, image resizing, etc), the MSB-bits also change, so that the cipher-images can not be decrypted back into the original images. In other words, encryption in spatial domain is not robust to common image processing.

In order to resistant to common image processing operations, the plain images should be encrypted in frequency domain. When the cipher image is modified, the modification only have little influence on low frequency elements, so that it can be decrypted into the plain images.

In this paper we propose an image encryption algorithm in frequency domain. The image is transformed by Discrete Cosine Transform (DCT). The previous research about image encryption in whole frequency domain has done and reported in [1]. Now we try to develop an block-based encryption algorithm. As we know DCT is used in JPEG compression process in which the original image is divided into blocks of size 8×8 pixel, and each block is transformed by DCT. In order to compatible to JPEG compression, in the proposed algorithm the original image is also divided into the such blocks. The DCT coefficients of each block is scrambled with an chaos-based permutation, i.e Arnold Cat Map. Arnold Cat Map is 2-D chaos map that transforms an element from a position to another position in the same area [2]. Because of DCT is lossy transformation, the proposed encryption algorithm is also lossy, that means the decrypted images are not exactly same as the original images. However, since on DCT domain, the encrypted images are robust to many image processing, such as JPEG compression, noising, etc.

II. BASIC THEORIES

2.1 Arnold Cat Map

Arnold Cat Map (ACM) is a two-dimensional chaotic map after discovered by Vladimir Arnold in 1960 [3]. Word "cat" appears because he uses the image of a cat in his experiments.

978-1-4799-7447-4/14/$31.00 ©2014 IEEE

ACM transform the coordinates (x, y) in the image of size $N \times N$ to the new coordinates (x', y'). The equation of ACM is

$$\begin{bmatrix} x_{i+1} \\ y_{i+1} \end{bmatrix} = \begin{bmatrix} 1 & b \\ c & bc+1 \end{bmatrix} \begin{bmatrix} x_i \\ y_i \end{bmatrix} \bmod(N) \tag{1}$$

where (x_i, y_i) is position of the pixel in the image, (x_{i+1}, y_{i+1}) is new pixel position after iteration i. Parameters b and c are any positive integer. Determinant of the matrix must be equal to 1 so that the transformation is area-preserving (remain in the same area of the image). ACM is iterated as m times and each iteration produces a random image. Values of b, c, and m can be considered as the secret keys.

The scramble image can be reconstructed into the original image using the same key (b, c, and m). The invers equation of ACM is

$$\begin{bmatrix} x_i \\ y_i \end{bmatrix} = \begin{bmatrix} 1 & b \\ c & bc+1 \end{bmatrix}^{-1} \begin{bmatrix} x_{i+1} \\ y_{i+1} \end{bmatrix} \bmod(N) \tag{2}$$

After last iteration, the original image is reconstructed succesfully.

2.2. Discrete Cosine Transform

The two dimensional DCT of an M by N matrix is defined as follows:

$$C(u,v) = \alpha_u \alpha_v \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} I(x,y) \cos\frac{\pi(2x+1)u}{2M} \cos\frac{\pi(2y+1)v}{2N} \tag{3}$$

and the inverse DCT (or IDCT) is given by

$$I(x,y) = \alpha_u \alpha_v \sum_{u=0}^{M-1} \sum_{v=0}^{N-1} C(u,v) \cos\frac{\pi(2x+1)u}{2M} \cos\frac{\pi(2y+1)v}{2N} \tag{4}$$

where

$$\alpha_u = \begin{cases} \frac{1}{\sqrt{M}} & ,u = 0 \\ \sqrt{\frac{2}{M}} & ,1 \le u \le M-1 \end{cases} \quad ; \quad \alpha_v = \begin{cases} \frac{1}{\sqrt{N}} & ,v = 0 \\ \sqrt{\frac{2}{N}} & ,1 \le v \le N-1 \end{cases}$$

The values $C(u, v)$ are called the DCT coefficients of image I. The upper leftmost element is called DC coefficient and the rest are called AC coefficients. The DCT can be applied to transform the whole image or image blocks (8×8 pixel).

III. The Proposed Algorithm

Withous loss of generalization for color images, let consider an grayscale image of size $N \times N$ pixels. The algorithm is simple. First, divide the plain-image into blocks of size 8×8 pixels and apply DCT for each bock. Next, scramble the DCT coefficients of each block, except DC element, by iterating Arnold Cat Map m times. The DC coefficient is not encrypted because it carries important visual information in an image. Finally, apply IDCT for each block to get the encrypted image (or the cipher-image).

If we run the simple algorithm above, the result is not so good, because the encrypted image still can be recognized (see Fig. 2, you can guess object in the image). This is result of local encryption of each block.

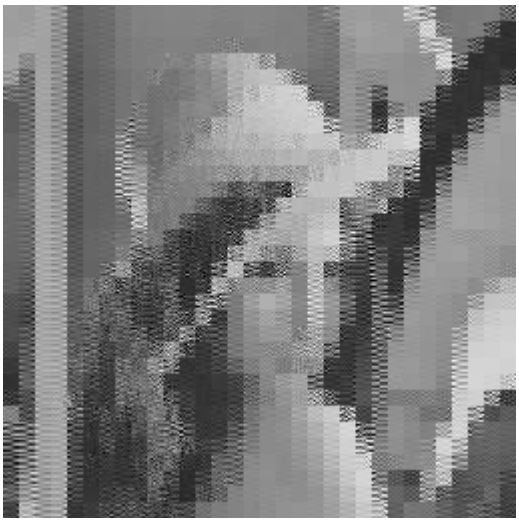


Fig.2.The encrypted image still can be recognized.

To overcome the weakness of the simple algorithm, we need to randomize the pixels of plain-image before we encrypt the DCT blocks. Thus, we use Arnold Cat Map twice, the first for scrambling the pixels of plain-image in spatial domain, and the second for scrambling the DCT coefficients of each block 8×8. Outline of the proposed selective encryption algorithm is as follows

1. Scramble the plain-image with Arnold Cat map (eq. 1).
2. Divide the scramble image into blocks of size 8×8
3. Apply DCT for each block (eq. 3).
4. Apply Arnold Cat Map to scramble the AC coefficients.
5. Apply IDCT to each block to get the cipher-image (eq. 4).

For decryption process, the process is same but in reverse order. Figure 3 shows each of encryption and decryption diagram of the proposed algorithm.

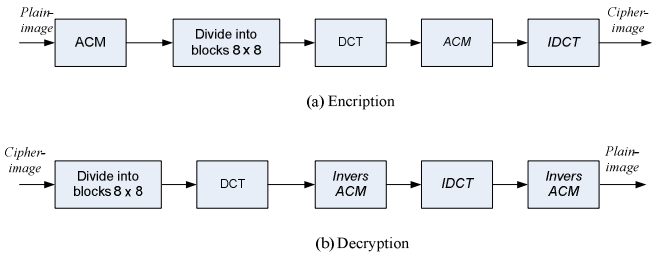


Fig 3. An encryption and decryption diagram of the proposed algorithm.

978-1-4799-7447-4/14/$31.00 ©2014 IEEE

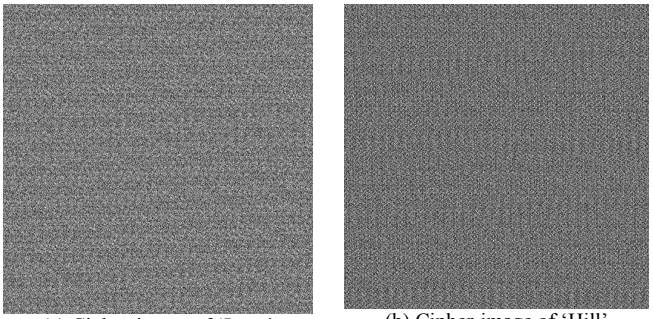
The secret keys of the algorithm are b, c, and m. Image encryption and decryption requires the same keys. Because of the DCT is a lossy transformation, then the image decryption does not yield exactly same as the original image. The image size must be square to ensure the implementation of Arnold Cat Map. If the size is not square then it needs additional pixels so that the image size is square.

IV. Experiment Results

Any grayscale images can be encrypted and then decrypted by the proposed algorithm above. Experiments are carried out by using Matlab tool. Two images that tested are 'Lena' and 'Hill' as shown Fig. 4(a) and 4(b)

(a) Lena (b) Hill

Fig 4. The plain images that used in experiments.



(a) Cipher-image of 'Lena' (b) Cipher-image of 'Hill'

(c) Decrypted image of 'Lena', PSNR = 32.4210 (d) Decrypted image of 'Hill', PSNR = 32.0223

Fig 5. (a) and (b) are plain-images; (c) and (d) are cipher-images; (e) and (f) are decrypted images.

Fig. 5 shows the cipher-images and the decrypted images. All images are 512 × 512 pixels. The secret keys are b = 32, c = 41, and m = 5. A PSNR (peak-signal-to-noise-ratio) is calculated by formula

$$PSNR = 20 \times \log_{10}\left(\frac{b}{rms}\right) \tag{5}$$

where b is peak signal (= 255 for grayscale image) and *rms* is an abbreviation of root mean square, i.e diference of two images I and $\hat{I}$,

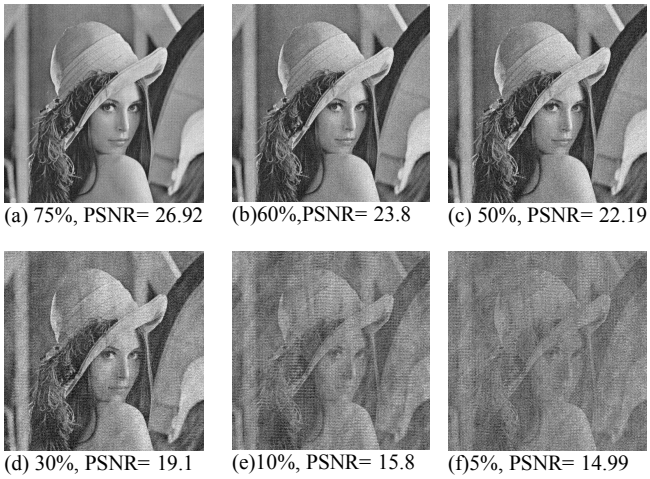
$$rms = \sqrt{\frac{1}{MN}\sum_{i=1}^{N}\sum_{j=1}^{M}(I_{ij} - \hat{I}_{ij})^2} \tag{6}$$

V. Robustness Analysis

Next we do some experiments to determine robustness of the cipher-images to common image processings. Such image processings are JPEG compression, image noising, image resizing. etc. Without generality, we test for 'Lena' image only. For some experiments we use MATLAB and for other experiments we use *Photoshop* software and discussion of the results refer to [1].

3.1. JPEG Compression

JPEG is common standard of image compression. JPEG compression causes the image size (in bytes) to be small and the image quality is reduced. In this experiment we save the cipher-images of 'Lena' of various compression quality into JPEG files using MATLAB code. The various compression qualities are 75%, 60%, 50%, 30%, 10%, and 5%. Fig.6 shows the decrypted images of 'Lena' after doing compression to the cipher-image. Quality of decrypted images (measured by PSNR) tends to decrease when quality of JPEG compression is reduced, but the decrypted image can be still recognized.



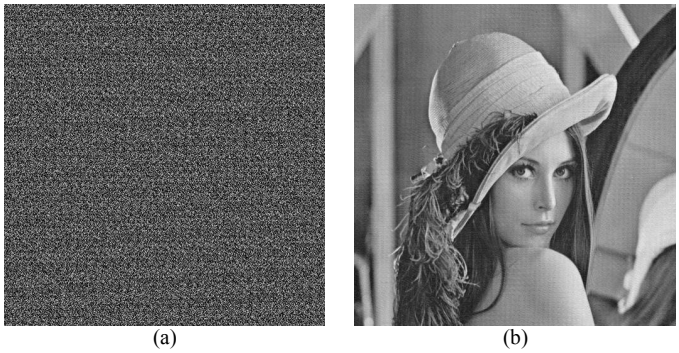
(a) 75%, PSNR= 26.92 (b)60%,PSNR= 23.8 (c) 50%, PSNR= 22.19

(d) 30%, PSNR= 19.1 (e)10%, PSNR= 15.8 (f)5%, PSNR= 14.99

Fig 6. The decrypted images of 'Lena' when the corresponding cipher-images is compressed by JPEG with various compression quality.

3.2. Brightness/Contrast Adjustment

Brightness or contrast adjustment is a common image manipulation. The manipulation changes intensity of pixels. In this experiment we use *Photoshop* software to adjust brightness and contrast of the cipher-image where the brightness is reduced to factor +30 and the contrast is increased to factor +30. Fig. 7 shows the cipher-image an the decrypted image after adjustment.Tthe decrypted image also adjusted well, since modification have only little influence on low frequency DCT coefficients.

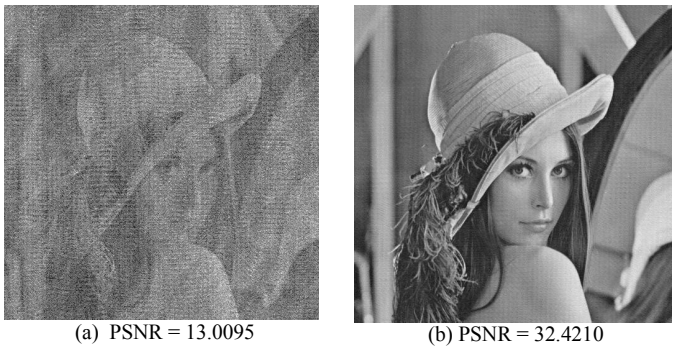


(a) (b)

Fig 7. (a) Cipher-image after doing brightness/contrast adjustment; (b) The decrypted image (PSNR = 32.4210)

3.3 Image Resizing

Image resizing is an operation to change the size of image. We can reduce or enlarge the size. In this experiment the cipher-image is resized by *Photoshop* software. In the first experiment we change the size of chipher-image of 'Lena' from 512×512 to 256×256 (reduction of), and in the second experiment we change from 512×512 to 1024×1024 (enlargement). After resizing, we resize it back to original size. Fig. 8 shows the decrypted images after resizing. For reduction of the size, the decrypted image contains a lot of noise, however the image can still be recognized. For enlargment case, the decrypted image has a good quality, same as before.



(a) PSNR = 13.0095 (b) PSNR = 32.4210

Fig 8. (a) The decrypted image after redunction of the size; (b) The decrypted image after enlargment

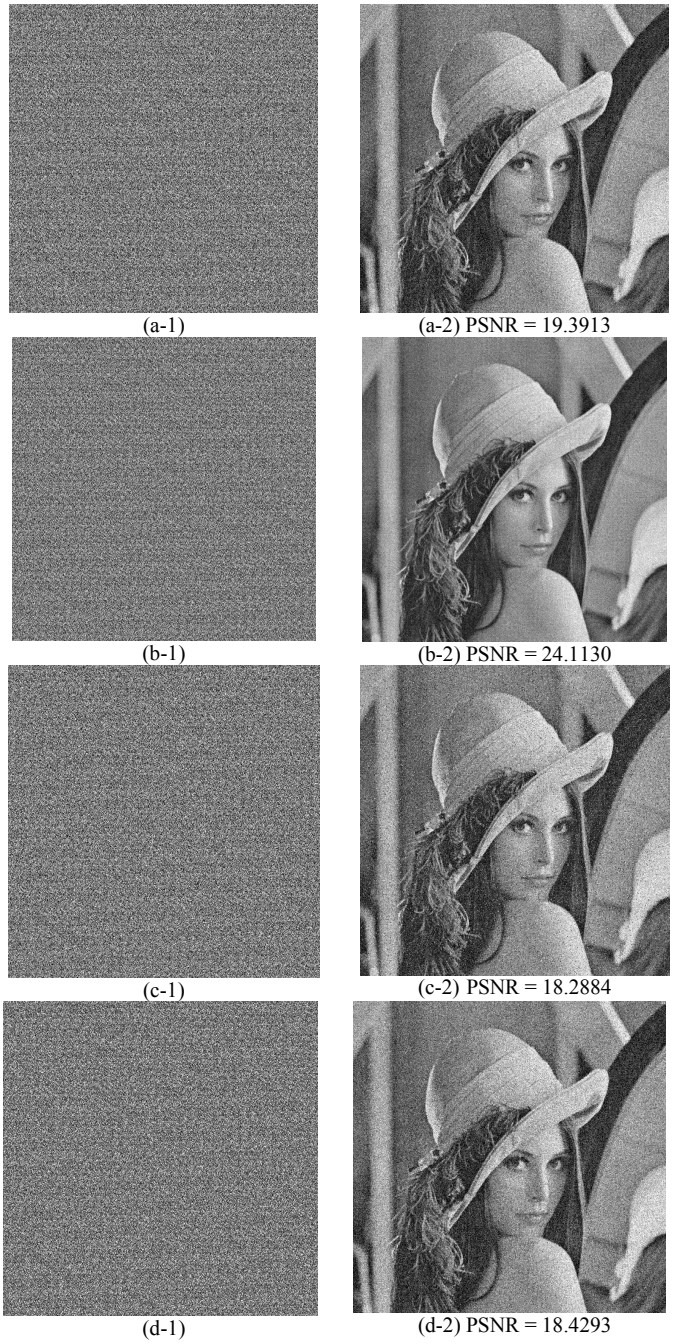
3.4 Image Noising

In these experiments we add different noise to the cipher-image using Matlab code:

```
J = imnoise(I,type)
```

Noise types are gaussian noise, poisson noise, salt and pepper noise, and speckle noise. For gaussian noise, a Matlab code is

```
J = imnoise(I,'gaussian',m,v)
```



(a-1) (a-2) PSNR = 19.3913
(b-1) (b-2) PSNR = 24.1130
(c-1) (c-2) PSNR = 18.2884
(d-1) (d-2) PSNR = 18.4293

Fig 9. (a-1) Gaussian noise added; (b-1) Poisson noise added; (c-1) Salt & pepper noise added; (d-1) Speckle noise added. The second images in the right column are corresponding decrypted images.

The default values is zero mean (`m`) noise with 0.01 variance (`v`). For poisson noise, a Matlab code is

978-1-4799-7447-4/14/$31.00 ©2014 IEEE

```
J = imnoise(I,'poisson')
```

and no default values. For salt and pepper noise, a Matlab code is

```
J = imnoise(I,'salt & pepper',d)
```

where `d` is the noise density. The default is 0.05 noise density. Finally, for speckle noise, a Matlab code is

```
J = imnoise(I,'speckle',v)
```

where `v` is the variance. The default for `v` is 0.04.

Fig. 9 shows the decrypted images after adding noise to the cipher-image. Overall the decrypted images contains noise, but the image can still be recognized.

VI. CONCLUSION

In this paper has presented a block-based image encryption algorithm using a chaotic permutation. Experiment shows that the cipher-images that resulted from the algorithm are robust to common image processing operations. Such image processing are JPEG compression, image noising, and image resizing. The decrypted images can be still recognized well, although they are just like noised.

REFERENCES

[1] Rinaldi Munir (2012), Robustness Analysis of Selective Image Encryption Algorithm Based on Arnold Cat Map Permutation, Proceeding of Makassar International Conference on Electrical Engineering and Informatics (MICEEI), 28 November-1 December 2012.

[2] Katherine Struss (2009): *A Chaotic Image Encryption*, Mathematics Senior Seminar, 4901, University of Minnesota, Morris.

[3] C. Wei-bin, Z. Xin (2009): *Image Encryption Algorithm Based on Henon Chaotic System*, Proceeding of International Conference on Image Analysis and Signal Processing (IASP 2009).

978-1-4799-7447-4/14/$31.00 ©2014 IEEE