2012 6th International Conference on Sciences of Electronics, Technologies of Information and Telecommunications (SETIT)

Projection Based Permutation of Color Images

Brahim NINI
Computer science department
University of Oum El-Bouaghi
Po. Box 358, 04000, Oum El-bouaghi, Algeria
Brahim_nini@yahoo.fr

***Abstract*—The encryption of images is based on two main complementary techniques: permutation and substitution of pixels. The strength of any encryption algorithm depends on the strength of both techniques. This paper presents an algorithm for rows and columns permutation of pixels. It introduces a virtual cylinder surrounding the image through which a virtual viewer looks at the image when rotating around it. The key idea is based on the fact that the line of sight of each pixel to the viewer intersects the cylinder surface at a given point. This point should be the same whatever the position of the viewer is. Therefore, it is used to find out the new pixel's position corresponding to the original image in a new generated image which is viewed from another position. The new created image has then some pixels which stack up and others which become out. They are then reintroduced in the created holes in the new image. This reinsertion of pixels creates the expected permutation. Based on only two main parts of the key which can be reinforced by others and a substitution technique, the algorithm shows a strength transformation of images for the purpose of their encryption.**

Key words-Image permutation, image transform, key, nonlinear permutation.

I. INTRODUCTION

It becomes commonly known that any important data exchange through a network should be subject to a previous encryption in order to avoid a misuse by any intruder. The particular case of images and videos attracts more attention due the bulk of direct interpreted information they hold. Unfortunately, a totally secure cryptographic algorithms are difficult to build because solutions can always be found to defeat any known algorithm. One way to ensure the efficiency of a cryptographic algorithm is that it must pass the test of time and general acceptance by cryptographic communities as shown in the selection of Advanced Encryption Standard (AES) [7]. In general, there are three major kinds of methods used for constructing secure encryption algorithms: permutation, substitution, and their combining form. The first kind is an image shuffling which makes the content hidden and confusing. The second one is the encryption of such content.

These kinds of algorithms have been extensively addressed in the literature and many developed systems have proved their efficiency. Most of them are based on chaotic systems [5] [6] [11] which is mainly used for the substitution process, though some critics are always done for an improvement purpose [4]. For the permutation process, a position permutation algorithm by magic cube transformation was for instance used in [5]. Like existing similar approaches, it is based on a transformation of magic cube's face values. It was [9] who proposed a new method for the construction of a magic cube through consecutive improvements. Furthermore, another approach is proposed in [8] based on combinations of hybrid magic cubes which are generated from a magic square and two orthogonal Latin squares. On another hand, a cryptographic algorithm which uses Maximum Distance Separable (MDS) matrices as part of its diffusion element is proposed in [3]. Also, [10] used matrix scrambling which is based on shifting and exchanging rule of bi-column bi-row circular queue.

In this paper, we propose a new idea of a color image permutation-only. It is based on a virtual cylinder having its diameter equal to the image size and centered on its middle. It is very close to the work presented in [1] with the difference that the positions the viewer takes are on a cylinder. The cylinder that surrounds the image plays the role of a mirror of an image through which it is viewed. It is assumed that the line of sight of any pixel in the image to the viewer crosses the cylinder surface at the same point whatever the position of the viewer is. So, based on the original view of the image, each point is used to estimate the position the corresponding pixel takes in a target image when viewed by a virtual viewer from another position. This may lead to a projection of the original image into the new one. What is noticeable in this case is that some pixels are projected onto the same positions, whereas others are completely excluded from the target image (Figure 1). As a result, the new reconstructed image contains many holes where no pixel is defined originating from the original one. The used method consists in refilling those holes by the excluded pixels and the ones which have the same positions with others because only one pixel is projected onto the same position to avoid information loss in the target image.

As already mentioned, this work is close to the one presented in [1]. The main important difference is in the positions the viewer takes. Whereas this one supposes the viewer taking her/his position round the central axis of the image, the one in [1] assumed these positions on a parallel line to the image plane. This leads to completely different mathematical expressions.

Even though our approach is based on a permutation-only algorithm, it holds some important features that make it closer

to a complete encryption method. In fact, it is based on few parameters which allow the creation of a cyphered image. Moreover, our algorithm becomes more powerful if supported by a substitution algorithm for encryption.

The rest of the paper is organized as follows. The next section details the principle of the virtual cylinder. It explains how the new image is generated. The third section explains how the scrambling of an image is obtained. The fourth section is interested in the experimental results. Finally, a conclusion is given summarizing the work and its possible extensions.

II. THE VIRTUAL CYLINDER PRINCIPLE

A virtual cylinder that surrounds an image is the pillar of the idea of our image permutation algorithm. A virtual viewer is supposed rotating round the image and looking at it through such virtual cylinder. This cylinder plays the role of a mirror transforming the image into a cylindrical form. Thus, it is assumed that, when looking at an image, the line of sight of each pixel crosses the surface of the cylinder at a specific point. This point's position of each pixel does not change when the location of the viewer changes. Based on this assumption and through the original image, it becomes possible to deduce any other view of the same image from another position. We define first the reference position in relation to the original image as being perpendicular to it and situated on its central axis. Second, it is assumed that any view of the image is on a perpendicular plane to the direction of the viewer (Figure 1). In other words, when the viewer goes round the image to takes a given position, as if the resulting image she/he views rotates about the cylinder's central axis to remain perpendicular to her/his direction. As a result, a new image can be generated by the projection of each original pixel to its relative position in the new image.

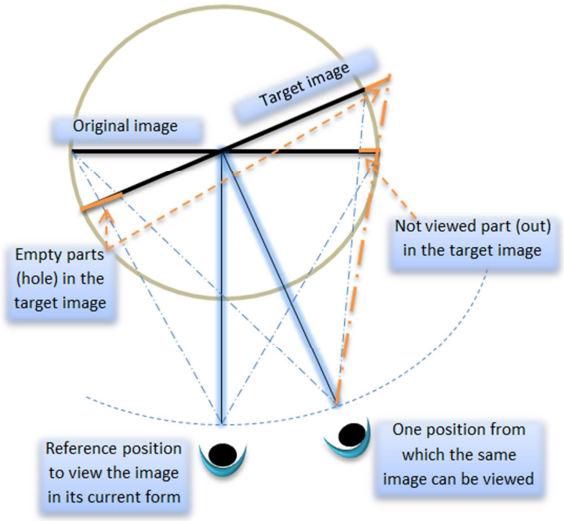


Figure 1. Transformation of the original image when viewed from different positions into a new one due to the virtual cylinder.

In order to understand how the new image is generated, and how the process of permutation is done, the next subsections state the mathematical basis that is the underneath of the overall process.

A. Mathematical basis of a particular position of the viewer

As stated before, the light of sight of any pixel in the image crosses the surface of the virtual cylinder. So, let us consider a pixel located at P_o from the middle of the image and viewed from the reference position situated at V_o (figure 2). This pixel has its image on the cylinder at the point P. The position in which it is viewed in the target (or generated) image depends on the new position of the viewer. It is supposed that the displacement to another position should be along the line of a circular direction round the central axis of the image. So, if a viewer is at a position V_t, the pixel is then projected on the point P_t in the new image. The objective is then to determine the amount of OP_t.

Let the angle between the directions of the reference position and a new one be θ. This angle is either directly known or calculated through the displacement $\overrightarrow{V_oV_t}$. Let also r be the radius of the virtual cylinder which is the image half size. Using the triangle ΔOP_oV_o and based on the fact that A is the orthogonal projection of P onto OV_o, it is easy to see that $AP = r.sin\,\lambda$, and $AV_o = OV_o - r.cos\lambda$. Moreover, the fraction $\frac{OP_o}{AP} = \frac{OV_o}{AV_o}$ being true, it becomes easy to obtain the expression of OP_o:

$$OP_o = \frac{OV_o.r.sin\lambda}{OV_o - r.cos\lambda} \tag{1}$$

Transformed into an equation where λ is the variable, expression (1) becomes:

$$\mathrm{OP_o}.\cos\lambda + \mathrm{OV_o}.\sin\lambda = \frac{\mathrm{OP_o}.\mathrm{OV_o}}{\mathrm{r}} \tag{2}$$

The solutions to equation (2) are those of (3):

$$\cos(\lambda - \varphi) = \frac{\mathrm{OV_o}.\cos\varphi}{\mathrm{r}} \tag{3}$$

where $tg\varphi = \frac{OP_o}{OV_o}$.

In the same way of reflection for the establishment of expression (1), the use of the triangle ΔOP_tV_t and the fact that PB is perpendicular to OV_t lead to the equality $\frac{OP_t}{BP} = \frac{OV_t}{BV_t}$ and the expression of OP_t:

$$OP_t = \frac{OV_t.r.sin\alpha}{OV_t - r.cos\alpha} = \frac{OV_o.r.sin\alpha}{OV_o - r.cos\alpha} \tag{4}$$

since $OV_t = OV_o$. Also, one can see that expressions (1) and (4) are similar. The only difference is the used angle.

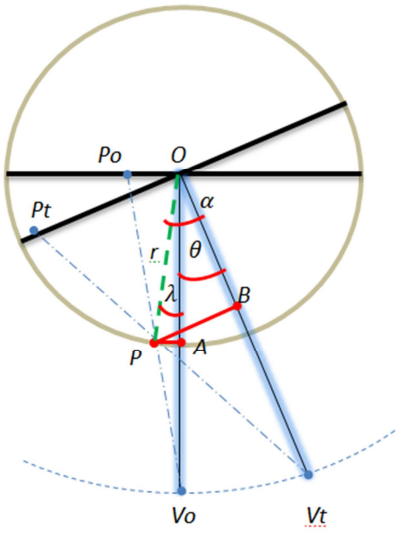
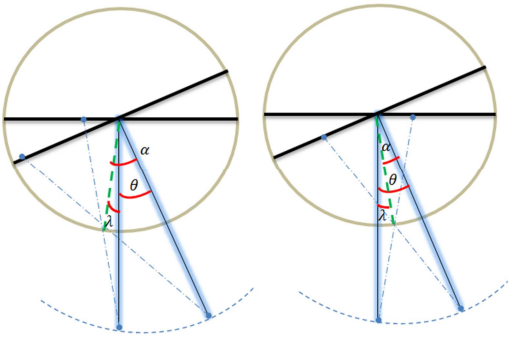


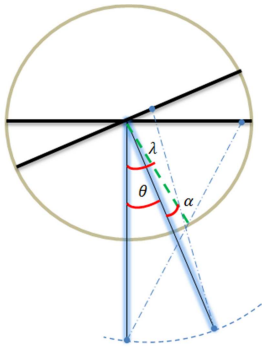
Figure 2. The system of projection that estimates the position of a pixel in the generated image.

Since α is linked to λ, the evaluation of OP_t in expression (4) depends on the solutions of equation (3). So, the value of λ which suits the described system is used to deduce the one of α. For this, note that the values of the three angles, θ, λ, and α, are taken positive, and particularly, the one of θ should be $0 < \theta < \pi/2$; otherwise, the image becomes no longer viewed.

Among the solutions of equation (3), the accurate is the one that fits the defined system. This means that α should be chosen so that its relation to λ remains always respected. It is clear that this relation depends on the position of the point P on the cylinder. In fact, there are three cases where α is expressed differently (figure 3). So, it is important to determine the boundaries in the original image that specify which expression of α should be used, and to select the part of the target image on which the pixel will be projected according to each expression.



$\alpha = \theta + \lambda$ $\alpha = \theta - \lambda$



$\alpha = \lambda - \theta$

Figure 3: Values of α depending on the position of the pixel and its projection on the cylinder.

Figure (3) shows that the junctions of OV_o and OV_t with the cylinder define the boundaries of different cases on both images. When the point of interest is the junction of OV_o with the cylinder, it means that $\lambda = 0$ and $\alpha = \theta$. This defines where the middle of the original image should be projected in the target one. Let us call it P_{t_0}(figure 4). In this case and according to expression (4), $OP_{t_o} = \frac{OV_o.r.sin\theta}{OV_o - r.cos\theta}$, and to find OP_t of any other pixel situated in the half left of the original image, the expression $\alpha = \theta + \lambda$ is used. So, any calculated value for OP_t should be larger than OP_{t_o}. Furthermore, the point which is the junction of $OV_t$and the cylinder is projected in the middle of the target image and its position in the original one is $r + OP_{o_t}$, where $OP_{o_t} = \frac{OV_o.r.sin\theta}{OV_o - r.cos\theta}$ according to expression (1). In fact, for such point, it is clear that $\lambda = \theta$. Consequently, for any pixel which is situated on the half right side of the original image and on the left side of OP_{o_t}, the expression $\alpha = \theta - \lambda$ is used for the sake to determine the value of α which will be used in expression (4). However, in the target image, the position the pixel takes is in the half left side. Therefore, any other pixel on the right of OP_{o_t} has its target position in the half right side of the target image, and to be determined, the expression $\alpha = \lambda - \theta$ is used.

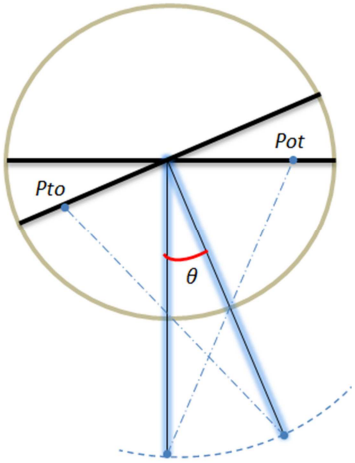


Figure 4. The positions where the middle of each image is positioned in the other one.

The established expressions are based on a viewer situated on the right side of the reference position. One can easily verify that every expression remains true for a mirror view. This means that if the viewer is on the left side, the reflection is simply done on the symmetric images to figures (2, 3, and 4). Consequently, any obtained result remains valid.

B. *New image generation*

In order not to consider every pixel alone, the idea considers the lines and columns of pixels as the unit of transformation. For this, the generation of the target image is based on the use of the previously established expressions and their application to a whole line or column of the image. It consists in evaluating the new position where every line or column in the original image should take in the target one. The parameters this evaluation considers depend on the position of the viewer in relation to the center of the original image. Hence, the transformation should be applied accordingly along the lines, columns, or both.

The transformation of the original image needs two basic parameters. They are the side the viewer takes from the reference position (angle θ) and her/his distance $OV_o = OV_t$. They constitute the most important parts of the permutation process key. Note that the distance and the angle are both formed by two components: the horizontal (OV_{t_x} and θ_x) and vertical (OV_{t_y} and θ_y) displacements. For each component, a vertical and a horizontal cylinder are respectively defined. Using these parameters, the algorithm of transformation of the original image is as follows:

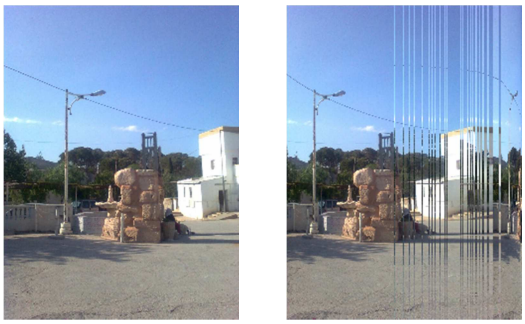
```
% Algorithm: New_Image_Gen
for each position (column/line) of
          the original image
  Evaluate the corresponding OPo;
  phi = atan(OVo/OPo);
  Solve equation (3) and chose
           the accurate value of lambda;
  Estimate OPt depending on position
           in relation to r and r+OPto
  if OPt is a defined position in
           the target image
    if OPt is not yet used
      Copy the column/line in OPo to the
          position OPt in the new image;
      Mark OPt as used;
    else
      Mark OPo as repeated and
              add it to P_rep;
    end
  else
    Mark OPt as out and add it to P_out;
  end
end
```

III. PERMUTATION OF IMAGE CONTENT

A. *Reintroduction of excluded pixels*

This process is the heart of the algorithm. In fact, there are two types of pixels which are excluded preliminary from the new generated image (see figure 1). The first type is constituted of pixels which are on the lines or columns that have their projected positions out of the boundaries of the image. They are called P_{out}. The second type includes pixels of several lines or columns having the same projected position. They are so, because any given position is used for the projection of only one line/column, and when the evaluated positions are rounded to their nearest integers, some of them become the same. They are called P_{rep}. Therefore, the second part of the proposed algorithm reintroduces simply these lines and/or columns so that they occupy the created holes into the generated image. Furthermore, in order to understand how these pixels are reintroduced, notice three important features in the projected image. The first one is that P_{out} are all situated on the limits of the image. They are close to each other and form a unique bloc which disappears completely from the image. The second one is that the created holes due to repetitions are naturally separated from each other. Finally, a bloc of undefined pixels appears on the opposite side of P_{out}. This is logic since it reflects the fact that as if the viewer is trying to look at some part of the image that does not exist.

Because the reintroduction should take into account the original positions in order to separate connected pixels from each other, the excluded pixels P_{out} are copied first. So, it is important not to reintroduce them in the opposite region of the image where there is no information. This is important, because this will be limited in a bloc exchange which reduces the capacity of shuffling in the permutation process. For this reason, P_{out} is introduced in the created holes inside the image because of P_{rep}. This splits the bloc into separated lines or columns, and hence, reinforces the shuffling process. To do it, the algorithm begins by the opposite side in the image of the viewer to search for existing holes in order to copy P_{out} into them.



(a) (b)

Figure 5. Application of ""New_Image_Gen" algorithm on an image with $OV_o = 1.5 \times Col$ and $\theta_x = \pi/12$ where (a) is the original image, and (b) its pojected and refilled form.

For P_{rep}, they are simply copied into the remaining holes. The analyzing of the positions of P_{rep} shows that concerned pixels (lines/columns) are always far from each other in the original image. Therefore, since they are initially not closer, they remain dispersed whatever the positions they are copied into in the target image. What is important, however, is that they should not be reintroduced in their initial positions. To ensure such criterion, they are reintroduced in the opposite order they have in the original image (Figure 5(b)).

B. Shuffling procedure

The target image is shuffled based on repeating the process of new image generation and reintroduction of excluded pixels several times in an intelligent way. In fact, several repetitions of the generation of a new image based on a previously generated one are enough to get a complete ciphered image. However, knowing the basis of this algorithm, it becomes easy to reconstitute the original image just by varying and combining different values of θ and OV_o. So, in order to make such operation difficult, the algorithm plays on many combinations of some parameters. In fact, there are several orders on which it is possible to play for the rearrangement of the target image. The first one is to apply "New_Image_Gen" algorithm alternatively along the columns and lines directions with different values of θ, i.e. θ_x and θ_y, and OV_o, i.e. OV_{t_x} and OV_{t_y}. The second one is to have for each step of a new generation different values of θ and OV_o, and alternate between them. Finally, the third one is to apply the rearrangement for RGB colors order too. These combinations make the retrieval of the original image very difficult because of the huge number of possibilities they create.

The overall algorithm becomes:

```
Image = Original_Image;
Repeat for a given number of permutation
  For each direction & alternatively do
    % direction may be lines (dir=y)
    % or columns (dir=x)
    Repeat for nbr_dir_repetition
      Select theta_dir and OVo_dir of
                the current step;
      Using New_Image_Gen and based on
                theta_dir and OVo_dir do
        Permute colors;
        Permute Image in the
                direction of dir;
      end
    end
  end
end
```

The permutation of colors from their initial order is made randomly, but based on θ and OV_o. The first step consists in transforming the vector $V_{order} = [0, 1, 2, 3, 4, 5]$ with the same algorithm "New_Image_Gen". Then, for each pixel, the modulo of the sum of its line and column to the value 6 is calculated. The aim is to select an order in which the three colors red, green, and blue are to be reordered. For this, the following orders are used:

0: do not change the current order
1: R B G
2: G R B
3: G B R
4: B R G
5: B G R

Therefore, the value of the modulo is used as the current index in V_{order}, and its current value is used to choose the new order to apply on the colors.

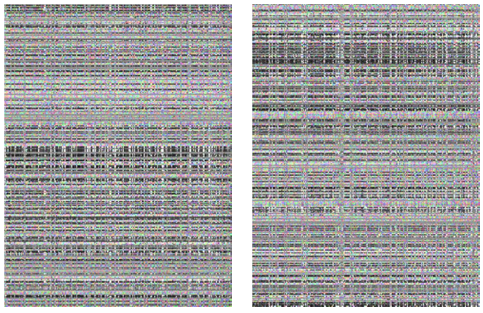
C. Reconstruction of the original image

The reconstruction of the original image is based on a reverse process of its permutation. Knowing the values of the first permutation, the reconstruction of the original image follows exactly the reverse steps. Simply, the algorithm repeats the same number of permutations, but at each step, it begins by the rearrangement of the colors then the line/columns of pixels.

IV. Experimental results and key analysis

A. Analysis of the permutation key

The permutation algorithm presented in this paper is strong enough to make the transformed image very difficult to reconstitute. In fact, the key of permutation may be as complex as we want. It depends on very simple values, but when combined together, they become very difficult to untie. For example, one form of the key may be $[400, 3, \frac{\pi}{8}, \frac{\pi}{12}, \frac{\pi}{9}, 1.5, 1.8, 1.3,$ $4, \frac{\pi}{20}, \frac{\pi}{15}, \frac{\pi}{19}, \frac{\pi}{11}, 1.7, 1.2, 1.4, 1.9]$ which is not so complex because it means that there will be 400 permutations, and at each one, the image is permuted 3 times along the columns using respectively the values $\left[\frac{\pi}{8}, \frac{\pi}{12}, \frac{\pi}{9}\right]$ for θ_x, and $[1.5, 1.8, 1.3]$ for OV_{o_x}, whereas it is permuted 4 times along the lines using the values $\left[\frac{\pi}{20}, \frac{\pi}{15}, \frac{\pi}{19}, \frac{\pi}{11}\right]$ for θ_y, and $[1.7, 1.2, 1.4, 1.9]$ for OV_{o_y}. The values of OV_{o_x} and OV_{o_y} are multiplied respectively by *nb_columns* and *nb_lines*. Note that at each step, the order of each pixel's colors in the image changes depending on its position and independently from the closest ones as explained before.



(a) (b)

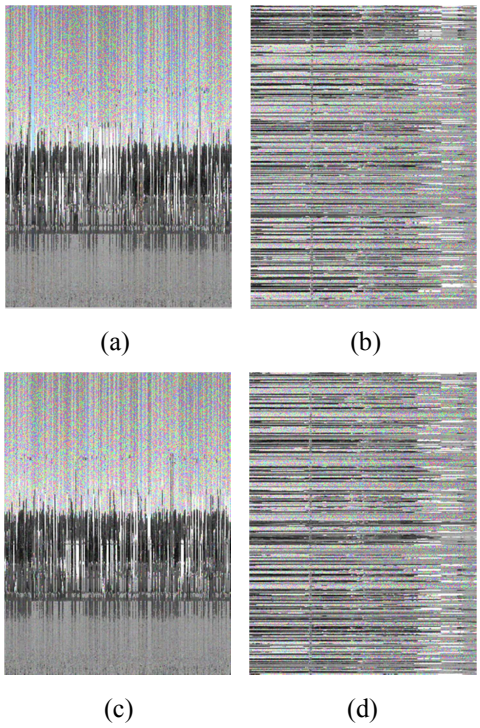
Figure 6. Permutation of the image in figure 5(a) using different values for the key. The values for (a) are: 200 iterations, $\theta_{x_1} = 4\frac{\pi}{45}$, $\theta_{x_2} = \frac{\pi}{18}$, $OV_{o_{x1}} = 1.5$, $OV_{o_{x2}} = 1.8$, $\theta_{y_1} = \frac{\pi}{15}$, $\theta_{y_2} = 17\frac{\pi}{180}$, $OV_{o_{y1}} = 1.7$, and $OV_{o_{y2}} = 1.1$, and the values for (b) are: 300 iterations, $\theta_{x_1} = 7\frac{\pi}{90}$, $\theta_{x_2} = \frac{\pi}{9}$, $OV_{o_{x1}} = 1.4$, $OV_{o_{x2}} = 1.0$, $\theta_{y_1} = \frac{\pi}{12}$, $\theta_{y_2} = \frac{\pi}{18}$, $OV_{o_{y1}} = 1.0$, and $OV_{o_{y2}} = 1.5$.

For instance, figure (6) shows the permutation of the image shown in figure (5(a)) with different parameters. The image shown in figure (6(a)) was shuffled using 200 permutations with two sub-iterations. The one shown in figure (6(b)) was shuffled using 300 permutations with also two sub iterations. What can be appreciable is that the number of permutations has no great effect on the result. From the appearance point of view, both images are completely blurred. The number of

iterations is important for the purpose to make the rearrangement difficult for any misuse.

B. Space values of the key

Depending on the initial values of the chosen key, the reverse process may be very sensitive. Just one value of θ which is different from the one used in the permutation process by 0.0001 does not retrieve the blurred image (Figures (7(a) and 7(b)). The retrieval of the original image is also affected by the values of OV_o; however, the sensitivity is about 0.001 (Figures (7(c) and 7(d)). So, given a 512×512 color image, if we consider that OV_o varies in the interval [0.5×512..2.5×512] = [256..1280] and θ in $\left[\frac{\pi}{180}..\frac{\pi}{4}\right]$ = [0.0175..0.7854], and for a number of iteration equal to 1000, the number of possible permutations is 1025000 (=(1280-256+1)×1000) × 7679000 (= (7854-175+1)×1000) ≈ 7.871×10^{12}. This number increases very fast when several sub-iterations are used. For example, the use of just two different values for θ_x and θ_y makes this number becoming 6.1952×10^{25}.



(a) (b)

(c) (d)

Figure 7 **:** Reconstruction of the image in figure (6(b)) having all values of the key equal to the original one except for (respectively from (a) to (d)): $\theta_{x_1} = 7\frac{\pi}{90} + 0.0001$, $\theta_{y_1} = \frac{\pi}{12} + 0.0001$, $OV_{o_{x1}} = 1.401$, and $OV_{o_{y1}} = 1.001$.

The size of the key depends on the chosen intended security for encryption. The more there are local permutations within the same step, the more the key size is bigger. The reason is that when several sub-permutations are used, θ and/or OV_o should use different values for each. This is why, the more the key is longer, the more the permutation becomes secure from attacks.

Moreover, together with an encryption algorithm, this permutation algorithm becomes more powerful. In fact, the reconstruction of the original image becomes almost impossible. Even though the used combinations of the different values of θ and/or OV_o may be deduced, the pixels themselves should be decrypted in order for the image to be understood. Moreover, the decryption process should substitute the correct pixels values in the correct order for the image to be viewed. This is why the number of combinations becomes larger than what was previously calculated.

C. Experimental results

Figures (8) and (9) shows the permutation process applied on the image used in [5] for encryption purpose. The result shows that when the colors are spread out in the image, the later becomes completely blurred without being encrypted (figure 9). In addition, the histograms of the two images show that, from statistical point of view, they are significantly different. Whereas in [5] the obtained histograms for the encrypted image are flat, the ones obtained in our case are not flat but very misguiding (histograms of figure 9). Note that this result is with no encryption involved.

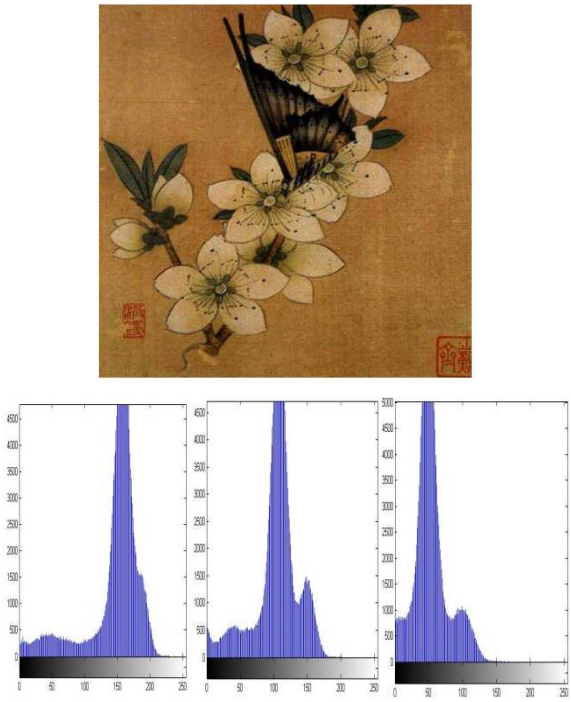


Figure 8. The same image used in [5] and its histograms (from left to right: histograms of red, green and blue planes).

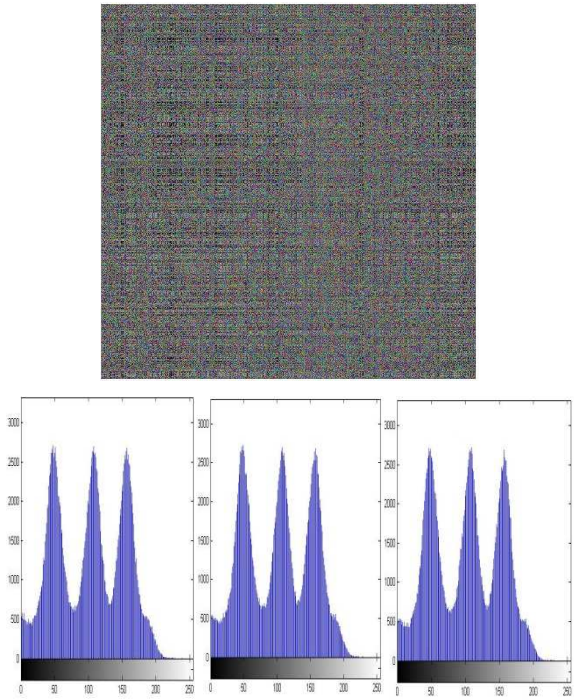


Figure 9. The same image in figure 8 permuted using 500 iterations, $\theta_{x_1} = 0.3491$, $\theta_{x_2} = 0.2443$, $\theta_{y_1} = 0.3316$, $\theta_{y_2} = 0.209$, $\theta_{y_3} = 0.209$, $OV_{o_{x1}} = 945$, $OV_{o_{x2}} = 1215$, $OV_{o_{y1}} = 776.1$, $OV_{o_{y2}} = 1014.9$, and $OV_{o_{y3}} = 1194$, and its histograms (from left to right: histograms of red, green and blue planes)

V. CONCLUSION

A new algorithm of an image permutation based on a geometrical nonlinear transform is presented. The algorithm is based on a virtual cylinder surrounding the image. Hence, for a viewer looking at the image from the reference position, s/he sees it as it is, and the line of sight of each pixel intersects the cylinder surface at a specific point. Therefore, when the viewer looks at the image from another position situated on a circular direction round it, each point is projected on a new perpendicular image to his/her direction. The new position the point takes is used to permute the pixel from its original position to this one. In the resulting image, many holes are then created due to some projected positions that are outside the target image and to other ones which become for several pixels when they are rounded the same. So, the pixels which are not directly permuted are reintroduced in the created holes. This process is repeated several times so that the entire image becomes blurred. This is what creates a permuted image of the original one.

The algorithm depends on the used key for shuffling that may make it strong enough to prevent any easy reconstruction of the original image. The core of the algorithm is based on the use of two parameters in the structure of the key: the angle of view and the distance from the image of the viewer. These two parameters may be duplicated many times, and at each iteration of the shuffling operation. Therefore, a huge number of possible combinations takes place which secures the reconstruction of the original image. What is also important in this algorithm is that several types of combinations can be used. It depends on the genuineness of the user.

The feasibility of our new algorithm has been shown through the sample image. Its sensitivity to randomly or planed generated keys is about 0.001 for the distance of the viewer and 0.0001 for the angle of view. This sensitivity may increase exponentially with the use of sub-parameters. Also, a permuted image shows that, from statistical point of view, its histograms are completely different from the original ones. This is why, it can be stated that the supporting of this algorithm by an encryption one may multiply its efficiency. In fact, our algorithm can be easily mixed to an encryption algorithm without any special transform. It is then considered as another layer in the encryption process which may be introduced at any step.

Further extensions of this work may be the study of the algorithm resistance to different attacks apart from the security considerations. This may lead to its improvement. For example, to what extent the permuted image may resist to JPEG compression in order to reconstruct the original one.

REFERENCES

[1] Brahim Nini, and Darine Bouteldja. Virtual Cylindrical View of a Color Image for its Permutation for an Encryption Purpose. *International Journal of Computer Applications,* Vol 16, No 1, pp. 11–17, February (2011).

[2] Chengqing Li.: On the Security of a Class of Image Encryption Scheme. *Proceedings of 2008 IEEE Int. Symposium on Circuits and Systems,* pp. 32903293, (2008).

[3] Daemen, J., and Rijmen, V.: The Design of Rijndael: AES. *The Advanced Encryption Standard*. Springer, (2002)

[4] Haojiang Gao, Yisheng Zhang, Shuyun Liang, and Dequn Li.: A new chaotic algorithm for image encryption. *Chaos, Solitons and Fractals*, 393-399, 29 (2006).

[5] Jianbing Shen, Xiaogang Jin, and Chuan Zhou.: A Color Image Encryption Algorithm Based on Magic Cube Transformation and Modular Arithmetic Operation. *Y.-S. Ho and H.J. Kim (Eds.): PCM 2005, Part II, LNCS* 3768, pp. 270-280, 2005. Springer-Verlag.

[6] Li. C., Li, S., Zhang, D., and Chen, G.: Cryptanalysis of a Chaotic Neural Network Based Multimedia Encryption Scheme., *Advances in Multimedia Information Processing PCM* 2004 Proceedings, Part III, LNCS, vol. 3333, pp. 418-425 (2004).

[7] National Institute of Standards (NIST): FIPS Pub 197: Advanced Encryption Standard AES. (2001), http://csrc.nist.gov/

[8] Sapiee Jamel, Tutut Herawan, and Mustafa Mat Deris.: A Cryptographic Algorithm Based on Hybrid Cubes, *ICCSA 2010, Part IV, LNCS* 6019, pp. 175187, 2010.

[9] Trenkler, M.: An Algorithm for making Magic Cubes. *The II ME Journal* 12(2), 105106 (2005)

[10] Wu, S., Zhang, Y., and Jing, X.: A Novel Encryption Algorithm based on Shifting and Exchanging Rule of Bi-Column Bi-row Circular Queue In: *International Conference on Computer Science and Software Engineering. IEEE*, Los Alamitos (2005)

[11] Zhang Linhua, Liao Xiaofeng, and Wang Xuebing.: An image encryption approach based on chaotic maps. *Chaos, Solitons and Fractals* 24 (2005) 759765.