Virtual Cylindrical View of a Color Image for its Permutation for an Encryption Purpose

Brahim Nini
Larbi Ben M'hidi University
Constantine Road, PO. BOX 358
04000 Oum El Bouaghi, Algeria

Darine Bouteldja
University M'hamed Bougara of Boumerdes, Algeria

ABSTRACT

This paper presents a novel algorithm for row and column permutation of pixels for the purpose of image encryption. The algorithm introduces a virtual cylinder surrounding an image and a virtual viewer looking at it but displaced from an original position. The key idea is based on the assumption that the light ray of each pixel to the viewer in her/his original position intersects the cylinder surface at a given point. When the viewer is displaced, the new position on a perpendicular image plane on which the pixel is projected should also have its direction intersecting the cylinder on the same point. As a result, all projected pixels in the new created image are slid from their original positions; but, some of them are delayed because they are piled up or projected out. In order to avoid information loss, these pixels are projected in the created holes of the new image. The consequence of such process is the creation of the expected permutation. Despite its simplicity, the algorithm shows a strong transformation of images for the purpose of their encryption.

General Terms

Image encryption, Security, Secured transfer, Network.

Keywords

Image permutation, Image transform, nonlinear permutation, Key.

1. 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) [5].

The strength of image encryption is based on the strength of the used techniques in permutation and substitution of pixels. 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 image's pixels shuffling which makes the content hidden and confusing. The second one is the encryption of the content of each pixel. 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 system [1, 2, 3] which is mainly used for the substitution process, although some critics are always done for an improvement purpose [9]. For the permutation process, a position permutation algorithm by magic cube transformation was for instance used in [2]. Like existing similar approaches, it is based on a transformation of magic cube's faces values. It was [8] who proposed a new method for the construction of a magic cube through consecutive improvements. Furthermore, another approach is proposed in [4] 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 [6]. Also, [7] uses 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 pixel-permutation. It is based on a virtual cylinder having its diameter equal to the image size and centered on its axis of symmetry. It is assumed that the light ray of any pixel in the image to the viewer crosses the cylinder surface at the same point whatever the position of the viewer is. Such assumption is based on another one which considers that the current view of the image is in a plane which is perpendicular to the position of the viewer. So, based on the original image, each point is used to estimate the position the corresponding pixel takes in a target image when viewed from a different position. This may lead to a projection of the original image onto a new one where all projected pixels are slid from their original positions. What is noticeable, however, is that some pixels cannot be directly projected. They are delayed because either they occupy the same positions in the target image, or they are completely excluded from it, because the positions they take are out of its boundaries (see Figure 1). As a result, the new constructed image contains many holes where no pixel originates from the original one. The used method consists in refilling those holes by the delayed pixels to avoid information loss. The process of projection-refilling creates our expected pixel-permutation.

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. The only missing operation in the encryption process is the substitution of pixels' values. So, when supported by a substitution algorithm for encryption, our algorithm becomes more powerful. Moreover, the mixture does not require any special changes in both algorithms.

The rest of the paper is organized as follows. The next section details the principle of a virtual cylinder. It explains the underneath mathematical basics and how the new image is generated. The third section goes more in detail to explain how the scrambling of an image is obtained. It gives the algorithms of permutation and the reverse processes. The fourth section is interested in the experimental results. It shows some obtained results, the analysis of the key and the sensitivity of the results. Finally, a conclusion is given summarizing the work and its possible extensions.

2. VIRTUAL CYLINDER PRINCIPLE

A virtual cylinder that surrounds an image is the pillar of the idea of our pixel-permutation algorithm. It is assumed that, when looking at an image, the light ray 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 position 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 having its view direction being perpendicular to the image plane and intersecting its axis of symmetry. Second, it is assumed that the viewed image from another position is on a perpendicular plane to the direction of the viewer as if the original image is rotated about the cylinder's axis. Consequently, a new image can be generated by the projection of each original pixel on its corresponding position in the new image. Note that the algorithm principle supposes that the displacement of the viewer is done along a perpendicular direction to the axis of the cylinder (see Figure 1).

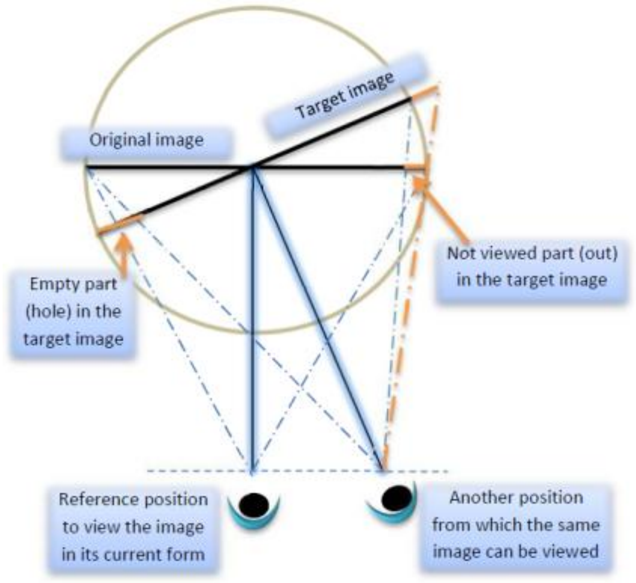

Target image
Original image
Not viewed part (out) in the target image
Empty part (hole) in the target image
Reference position to view the image in its current form
Another position from which the same image can be viewed

Fig 1: Transformation of the original image when viewed from different positions due to the virtual cylinder.

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

2.1 Mathematical Expressions of a Particular Position of the Viewer

As stated before, it is assumed that the light ray of any pixel in the image crosses the surface of the 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 (see Figure 2). This pixel has its light ray which is intersecting 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. So, if a viewer is at a position V_t, the pixel is projected on the point P_t in the new image. The objective is then to determine the amount of OP_t.

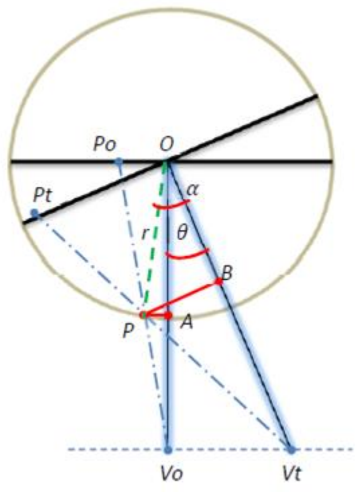


Fig 2: The system of projection that estimates the position of a pixel in the generated image.

Let the angle between the direction of the reference position and a new one be μ. This angle is either directly known or calculated through the displacement $\overrightarrow{V_oV_t}$. Let also r be the radius of the cylinder which is the image half size. Let now θ be $\alpha - \mu$ for the case of figure (2). Using the $\triangle\, OP_oV_o$ and based on the fact that A is the orthogonal projection of P onto OV_o, it is easy to see that $AP = r \cdot \sin\theta$, and $AV_o = OV_o - r \cdot \cos\theta$. Moreover, the fraction $\frac{OP_o}{AP} = \frac{OV_o}{AV_o}$ being true, it becomes easy to obtain the expression of OP_o:

$$OP_o = \frac{OV_o \cdot r \cdot \sin\theta}{OV_o - r \cdot \cos\theta} \qquad (1)$$

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

$$OP_o \cdot \cos\theta + OV_o \cdot \sin\theta = \frac{OP_o \cdot OV_o}{r} \qquad (2)$$

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

$$\cos(\lambda - \varphi) = \frac{OV_o \cdot \cos\varphi}{r} \qquad (3)$$

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

In the same way of reflection for the establishment of expression (1), the use of $\triangle OP_tV_t$ 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:

$$OP_t = \frac{OV_t \cdot r \cdot \sin\alpha}{OV_t - r \cdot \cos\alpha} = \frac{OV_o \cdot r \cdot \sin\alpha}{OV_o - r \cdot \cos\alpha \cdot \cos\theta} \qquad (4)$$

since $OV_t = OV_o / \cos\theta$.

As α is linked to λ, the estimation 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 (see Figure 3). So, it is important to determine in the original image the ranges that specify which expression of α should be used, and to select the part of the target image on which the pixel should be projected according to each expression.

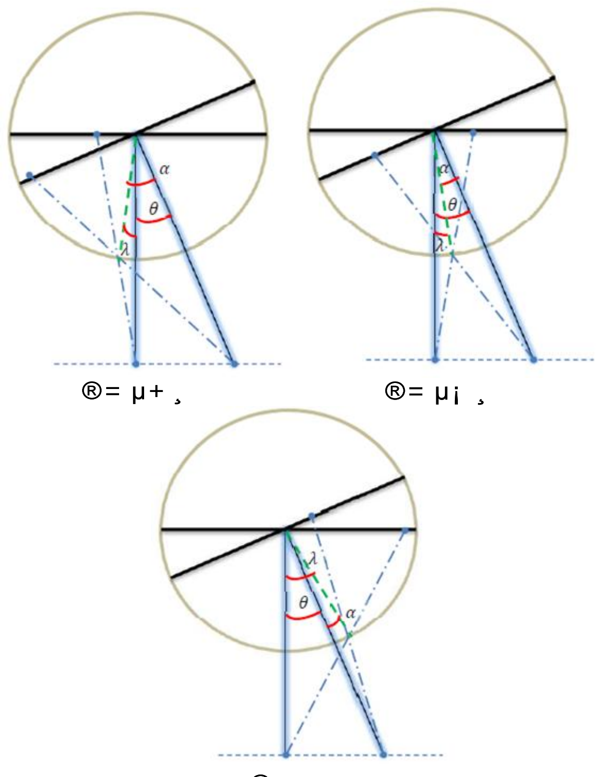

α
θ
λ
$\alpha = \theta + \lambda$
$\alpha = \theta - \lambda$
$\alpha = \lambda - \theta$

Fig 3: Values of α depending on the position of the pixel and its image 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_o} (see Figure 4). In this case and according to expression (4), $OP_{t_o} = \frac{OV_o \cdot r \cdot \sin\theta}{OV_o - r \cdot \cos^2\theta}$, and to find OP_t of any other pixel situated on the left half of the original image, the expression $\alpha = \theta + \lambda$ is used. So, any calculated value for OP_t is 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 $OP_{o_t} = \frac{OV_o \cdot r \cdot \sin\theta}{OV_o - r \cdot \cos\theta}$ according to expression (1). In fact, for such point, it is clear that $\lambda = \theta$. Consequently, for any pixel which is 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 is used in expression (4). However, in the target image, the position the pixel takes is in the left half 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.

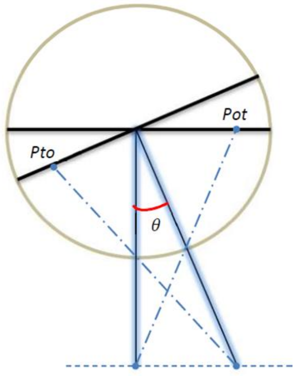


Fig 4: Where the middle of each image is positioned in the other one.

The established expressions are all based on the fact that the viewer is 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 symmetrical images to figures (2, 3, and 4). So, any obtained result remains valid.

2.2 New Image Generation

In order not to consider every pixel alone, the generation of the new image considers a whole line/column of pixels as the unit of transformation. Furthermore, in order to apply transformations along the columns and lines, there is a need to consider respectively vertical and horizontal cylinders, and for both, the projection is based on the use of the previously established expressions ((1) to (4)). So, along each direction, the new position where every line or column in the original image should take in the new one is evaluated. The parameters this transformation considers depend then on the position of the viewer in relation to the axis of symmetry of the original image.

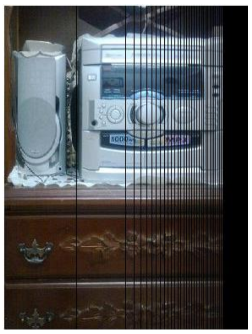
The transformation of the original image needs then two basic parameters. They are the side the viewer takes from the reference position (angle μ) and her/his distance OV_o from the image which is used to determine OV_t. They constitute the most important parts of the permutation process key. Therefore, since the permutation is done along horizontal and vertical directions and requires two virtual cylinders, the distance OV_t and the angle μ are both formed by two components: the horizontal (OV_{o_x} and μ_x) and vertical (OV_{o_y} and μ_y) displacements.

In addition, there are two types of pixels which are not preliminary projected onto the new generated image (see Figure 1). The first type is constituted of lines or columns that have their projection positions out of the boundaries of the image. They are called P_{out}. The second type includes the several lines or columns having the same projection positions. They are so, because the evaluated positions are rounded to their nearest integers, and some of them become the same. Consequently, any given position is used for the projection of only one line/column, and the others are delayed. The latter are called P_{rep}. Thus, because not all lines/columns are projected, some holes appear in the new image where no information is provided from the original one.

Based on these considerations, the algorithm of initial transformation of the original image is as follows:

```
% Algorithm: New_Image_Gen
for each position (column/line) of the original image
  Evaluate the corresponding $OP_o$;
  phi = atan($OV_o$/$OP_o$);
  Solve equation (3) and chose the accurate value of lambda;
  Estimate $OP_t$ depending on the position in relation to r and r+$OP_{to}$;
  if $OP_t$ is a definite position in the target image
    if $OP_t$ is not yet used
      Copy the column/line at $OP_o$ to the position $OP_t$ in the new image
      Mark $OP_t$ as used;
    else
      Mark $OP_o$ as repeated and add it to $P_{rep}$;
    end
  else
    Mark $OP_o$ as out and add it to $P_{out}$;
  end
end
```

(a) Original


(b) Transformed

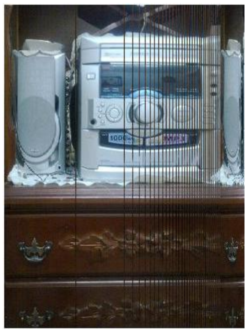

(c) Transformed and filled

Fig 5: Application of "New_Image_Gen" algorithm on an image with $OV_o = 1{:}5£ Col$ and $\mu_x = ¼{=}12$

3. IMAGE CONTENT PERMUTATION

3.1 Reintroduction of Delayed Pixels

The second part of the proposed algorithm simply reintroduces the delayed lines and/or columns so that they occupy the created holes into the generated image. So, 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 border of the image. They are contiguous and form a unique bloc which disappears completely in the new image. The second one is that the lines/columns that are projected on the same positions and delayed are physically dispersed. Finally, a set of dispersed and undefined lines/pixels appears on the opposite side of the viewer because this part becomes stretched.

For the reason that the reintroduction should take into account the original positions in order to separate contiguous pixels from each other, P_{out} are introduced first. As it is clear in figure (5(b)), the large hole is on the same side of the viewer. So, it is important not to reintroduce P_{out} into it. 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 of the stretched part. This splits the bloc into separated lines or columns, and hence, reinforces the shuffling process. To do it, the algorithm begins by searching from the opposite side of the viewer for existing holes in order to paste P_{out} into them.

For P_{rep}, they are simply pasted into the remaining holes. Since they are initially not close to each other, they remain dispersed whatever the positions into which they are pasted. Moreover, they are reintroduced in the new image in a reverse order from their initial one (see Figure 5(c)).

3.2 Shuffling Procedure

The target image is shuffled based on repeating the process of new image generation and reintroduction of delayed pixels several times. 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 basics of this algorithm, it becomes easy to reconstitute the original image just by varying and combining different values of μ and OV_o. So, in order to make such operation difficult, the algorithm combines the parameters in different ways. 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 μ and OV_t,. The second one is to use for each step of a new generation different values of μ and OV_t, and alternate between them. Finally, the third one is to apply the rearrangement to RGB colors order too. These combinations make the retrieval of the original image very difficult because of the huge number of possibilities they create.

Furthermore, the permutation of colors from their initial order is made randomly, and based on μ and OV_0. 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 order to apply on the colors.

The overall algorithm becomes:

```
Image = Original_Image;
Repeat for a given number of permutation
  For each direction do alternatively
    % 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 do
        Permute color bytes;
        Permute Image in the direction of dir;
      end
    end
  end
end
```

3.3 Reconstruction of the Original Image

The reconstruction of the original image is based on a reverse process of its permutation. Knowing the used values of μ and OV_0 during the first permutation, the reconstruction of the original image follows exactly the reverse steps.

4. KEY ANALYSIS AND EXPERIMENTAL RESULTS

4.1 Structure of the Permutation Key

The permutation algorithm which is 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 it is expected. 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 $[400; 3; \frac{¼}{8}; \frac{¼}{12}; \frac{¼}{9}; 1{:}5; 1{:}8; 1{:}0; 4; \frac{¼}{20}; \frac{¼}{15}; \frac{¼}{19}; \frac{¼}{11}; 1{:}7; 1{:}2; 1{:}4; 1{:}9]$ which is not so complex. 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 $[\frac{¼}{8}; \frac{¼}{12}; \frac{¼}{9}]$ for μ_x, and $[1{:}5; 1{:}8; 1; 3]$ for OV_{0_x}, whereas it is permuted 4 times along the lines using the values $[\frac{¼}{20}; \frac{¼}{15}; \frac{¼}{19}; \frac{¼}{11}]$ for μ_y, and $[1{:}7; 1{:}2; 1{:}4; 1{:}9]$ for OV_{0_y}. The values of OV_{0_x} and OV_{0_y} are multiplied respectively by $nb\_columns$ and nb_lines which are respectively columns and lines numbers of the image. Note that at each step, the order of each pixel's colors in the image changes depending on its position.

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)) is shuffled using 200 permutations with two sub-iterations. The one shown in figure (6(b)) is shuffled using 300 permutations with also two sub iterations. What is remarkable is that the number of permutations has no great effect on the result. From the appearance point of view, both images are completely non-comprehensible. The number of iterations is important for the purpose to make the rearrangement difficult for any misuse.

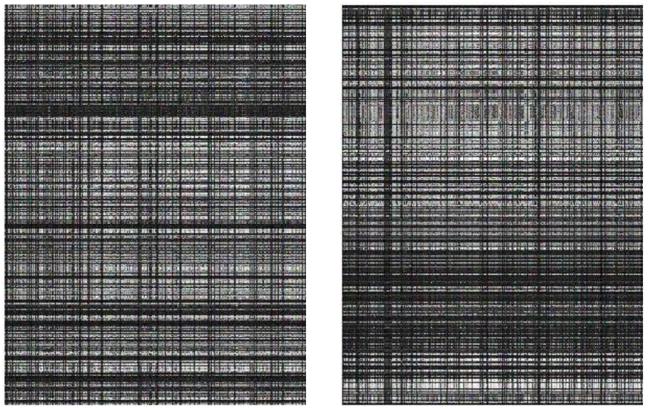

(a) (b)

Fig 6: Permutation of the image in figure 5(a) using different values for the key. The values for (a) are: 200 iterations, $\mu_{x1} = 16\frac{¼}{180}$, $\mu_{x2} = 10\frac{¼}{180}$, $OV_{x1} = 1{:}4$, $OV_{x2} = 1{:}8$, $\mu_{y1} = 12\frac{¼}{180}$, $\mu_{y2} = 17\frac{¼}{180}$, $OV_{y1} = 1{:}7$, and $OV_{y2} = 1{:}1$, and the values for (b) are: 300 iterations, $\mu_{x1} = 14\frac{¼}{180}$, $\mu_{x2} = 20\frac{¼}{180}$, $OV_{x1} = 1{:}4$, $OV_{x2} = 1{:}0$, $\mu_{y1} = 15\frac{¼}{180}$, $\mu_{y2} = 10\frac{¼}{180}$, $OV_{y1} = 1{:}0$, and $OV_{y2} = 1{:}5$.

4.2 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 μ which is different from the one used in the permutation process by 0.0001 blurs the retrieved image (see Figures 7(a) and 7(b)). The retrieval of the original image is also affected by the values of OV_0, however the sensitivity is about 0.001 (see Figures 7(c) and 7(d)). So, given a RGB $512 £ 512$ color image, if we consider that OV_0 varies in the interval $[0{:}5 £ 512{:}{:}2{:}5 £ 512] = [256{:}{:}1280]$ and μ in $[pi{=}180{:}{:}pi{=}4] = [0{:}0175{:}{:}0{:}7854]$, and for a number of iterations equal to 1000, the number of possible permutations is $1025000 (= (1280 ¡ 256 + 1) £ 1000) £ 7680000 (= (7854 ¡ 175 + 1) £ 1000) ¼ 7{:}872 £ 10^{12}$. This number increases rapidly when several sub-iterations are used. For example, the use of just two different values for μ_x and μ_y makes this number becoming $6:1968 £ 10^{25}$, because the same number of combinations is obtained on both directions, i.e., horizontal and vertical.

The size of the key depends on the chosen intended security for the permutation. The more there are local permutations within the same step, the more the key size is bigger. However, 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 μ 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.

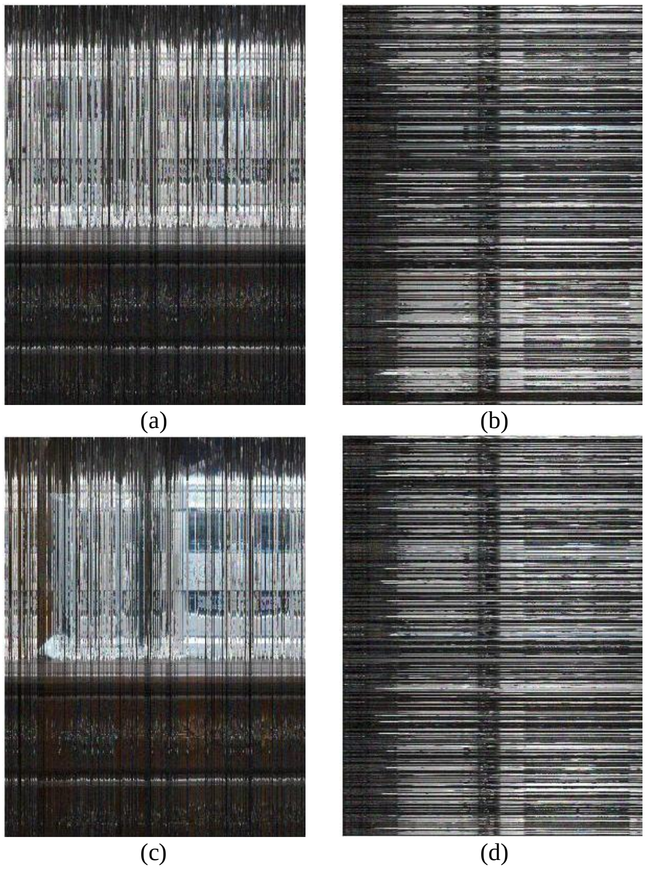


Fig 7: Reconstruction of the image in figure 6(b) having all values of the key equal to the original one except for (respectively from (a) to (d)): μ_x $(\mu_{x1_{used}} = 14\frac{¼}{180} + 0:0001)$, μ_y $(\mu_{y1_{used}} = 15\frac{¼}{180} + 0:0001)$, OV_{o_x} $(OV_{o_{x1_{used}}} = 1:5 + 0:001)$, **and** OV_{o_y} $(VO_{y1_{used}} = 1:0 + 0:001)$

4.3 Experimental Results

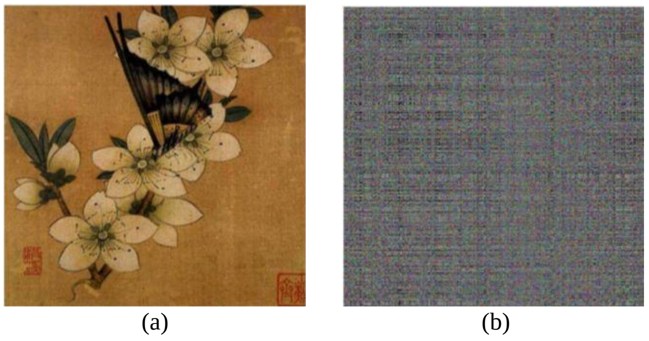


Fig 8: The same image used in [2] permuted using 500 iterations, $\mu_{x1} = 0:3491$, $\mu_{x2} = 0:2443$, $\mu_{y1} = 0:3316$, $\mu_{y2} = 0:2094$, $\mu_{y3} = 0:2618$, $OV_{o_{x1}} = 945$, $OV_{o_{x2}} = 1215$, $OV_{o_{y1}} = 776:1$, $OV_{o_{y2}} = 1014:9$, **and** $OV_{o_{y3}} = 1194$.

Figure (8(b)) shows the permutation process applied on the same image (Figure 8(a)) used in [2] for an encryption purpose. The result shows that when the colors are spread out in the image, the latter becomes completely blurred without being encrypted. In addition, the histograms of the two images show that, from statistical point of view, they are significantly different. Whereas, in figure (8(b)), the obtained histograms in [2] for the encrypted image are flat, the ones obtained in our case are not, but very misguiding. Note that this result is with no encryption involved.

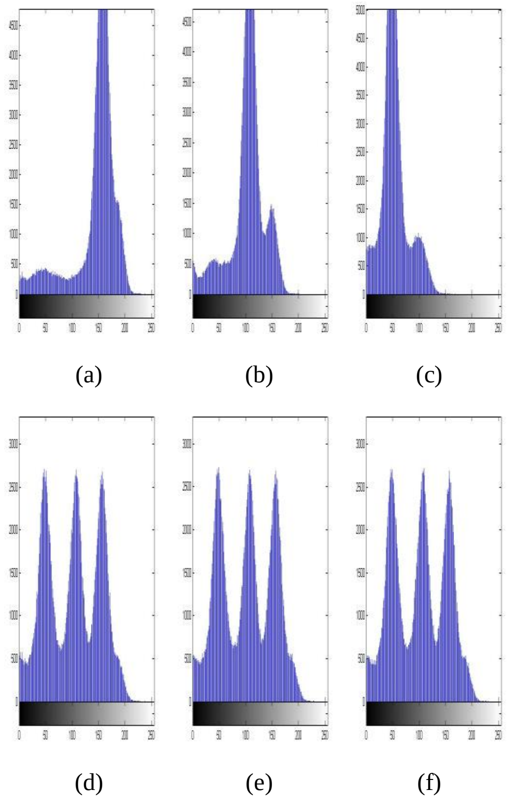


Fig 9: Histograms of the two previous images: (a), (b), and (c) of 8(a) and (d), (e), and (f) of 8(b).

5. CONCLUSION

A new algorithm of an image permutation based on a geometrical projection 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 light ray of each pixel intersects the cylinder surface at a specific point. Therefore, when the viewer looks at the image from another position, each point is projected on a new perpendicular image to her/his 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 the part which becomes stretched. 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 heavily on the used key for shuffling. Its structure is able to make it strong enough to prevent any easy reconstruction of the original image. The core of the key is based on the use of two parameters: the angle of view and the distance from the image of the viewer. These two parameters may be used with different values at each step of the shuffling operation. Therefore, a huge number of possible combinations take 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 algorithm has been demonstrated. Its sensitivity to random or planed generated keys is about 0.001 for the distance of the viewer and 0.0001 for the angle of view. This sensitivity increases rapidly 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, and this without any special transformation. It can be considered as another layer in an encryption process.

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.

6. REFERENCES

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

[2] Jianbing Shen, Xiaogang Jin, and Chuan Zhou. 2005. 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.

[3] Zhang Linhua, Liao Xiaofeng, and Wang Xuebing. 2005. An image encryption approach based on chaotic maps. Chaos, Solitons and Fractals 24, pp.759-765.

[4] Sapiee Jamel, Tutut Herawan, and Mustafa Mat Deris. 2010. A Cryptographic Algorithm Based on Hybrid Cubes. ICCSA 2010, Part IV, LNCS 6019, pp. 175-187.

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

[6] Daemen J., and Rijmen V. 2002. The Design of Rijndael: AES – The Advanced Encryption Standard. Springer-Verlag.

[7] Wu S., Zhang Y., and Jing X. 2005. 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.

[8] Trenkler, M. 2005. An Algorithm for making Magic Cubes', The Π ME Journal, Vol. 12, No 2, pp.105-106.

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

[10] Chengqing Li. 2008. On the Security of a Class of Image Encryption Scheme. In Proceedings of 2008 IEEE Int. Symposium on Circuits and Systems, pp. 3290-3293.