Bit-projection Based Color Image Encryption using a Virtual Rotated View

Brahim Nini
University of Oum El-bouaghi, Po. Box 358, 04000. Algeria. Tel. +213 661 487 057
E-mail: b.nini@univ-oeb.dz

Keywords: image encryption, bit permutation, pixel substitution, projection, security

Received: April 10, 2013

This paper presents a novel algorithm for a color image encryption which involves simultaneously two operations in one: permutation and substitution of pixels. It uses the rows and columns of images' bits as transformation units. Each bit is regarded as an observed entity having a light ray which intersects a new rotated image. The intersection position is used to paste the corresponding bit. Such projection makes the pixels' bits migrate between each other which generate a cipher image. Despite its simplicity, the algorithm shows a great resistance against many kinds of attacks through its sensitivity to the initial values used in encryption.

Povzetek: Opisan je nov algoritem za kodiranje barvnih slik.

1 Introduction

It becomes commonly known that any important data exchange, mainly through a network, should be subject to a previous encryption in order to avoid its misuse. The particular case of images and videos attracts more attention due to the bulk of direct interpreted information they hold.

There are three major kinds of methods used to construct secure encryption algorithms: permutation, substitution, and their combining form. The first kind, called in other works confusion, is an image shuffling which makes the content hidden and confusing. The second one, called in other works diffusion, is the coding of all pixels' values by new values instead of the original ones. The aim of both techniques, either used separately or combined, is to make the retrieval of the original image by attackers either impossible or very difficult. Unfortunately, totally secure cryptographic algorithms are difficult to build because solutions can always be found to defeat them.

Many kinds of algorithms have been extensively addressed in the literature. Recently, chaotic systems have been widely used in data encryption [5, 6, 7, 16, 17, 18]. Many ideas are proposed. The one presented in [6], for instance, is based on image permutation and proposes the scrambling/permuting binary bit of every pixel with pseudo-random number sequence generated by chaotic logistic map. Another method is proposed in [5]. It is a cryptographic algorithm which uses Maximum Distance Separable (MDS) matrices as part of its diffusion element. From another point of view, many other kinds of algorithms are proposed [8, 9, 11, 14, 15]. For instance, a position permutation algorithm by magic cube transformation for the permutation process is used in [8]. It is based on a transformation of magic cube's face values. It was proposed by [14] who improves it through consecutive work. Furthermore, another approach is proposed in [11] which is based on combinations of hybrid magic cubes which are generated from a magic square and two orthogonal Latin squares. Likewise, a matrix scrambling is used in [15]. It is based on shifting and exchanging rule of bi-column bi-row circular queue. In sum, several ideas are used for the purpose of image encryption algorithms development and each one has its advantages and drawbacks. In the reference [19], one may find out many recent advances in the field through the presented survey.

Though the quality of many works, results improvement is always required. For this, some works are oriented to analysis and/or refining of existing algorithms [2, 3, 4, 10, 12, 13]. These works focus on finding out weaknesses of developed methods and the way they can be strengthen. In [12] for instance, some means are proposed to strengthen the overall performance of the security of Fridrich's algorithm. Though there is no general model for security analysis suitable for all cryptosystems using different methods, some means are used to measure the strength of security and the speed of encryption process. They are based on the study of the behavior of encryption algorithms against attacks. The key and encrypted content are then analyzed whether they are safe or not against brute force, statistical, known plaintext, select plaintext, differential, and other attacks.

Following the domain's stream, this work proposes a new idea of color image encryption extending a previous one [1]. The latter was limited to an image permutation for its encryption purpose; but this one generalizes it to image encryption. It merges both permutation and substitution of pixels in one indiscernible operation.

In this work, an image is considered as rows and columns of bits. Each bit is regarded as an observed entity having a light ray. On another hand, a virtual viewer, who is assumed revolving about the middle axis of the image, involves a new view of the image that looks as if it is rotated so that its plane is perpendicular to the direction of the view. Such view is used as a new image onto which a projection is done. Accordingly, the intersection of a virtual bit's light ray to the viewer with the rotated image is used as the position of projection. This makes the bits moving within different pixels.

Since the same idea of projection in the former work [1] is used, the same effects are met. The projection process leads to some parts of the original image having their projection positions either out of bounds or being the same. The former case is evident due to the image plane surface limits. The latter is so, because the computed positions should be rounded to their nearest integers, and hence, some bits may have the same target positions. So, in order to avoid information loss, only one bit is projected in each position and the others are delayed together with out-projected bits. As a result, the new reconstructed image contains many holes where some positions are undefined, i.e., not originating from the original image. The idea of the proposed algorithm consists in refilling these holes by the delayed bits in a reverse order.

Our algorithm is nearly close in its function to the one proposed in [9] in that it repeats the permutation process several times with different values of the key. The difference is that, in our case, the key does not change dynamically during the permutation process, but might have different combining forms.

Moreover, the proposed algorithm holds some important features that make it a powerful encryption technique. This is proved through some measures based on some means proposed in [12] and detailed later. Moreover, this algorithm's features are challenging comparatively to other works in the following ways:

– The algorithm is based on a key whose parts are the parameters of a virtual viewer's position. Hence, the values of the key are understandable and measurable physical quantities,

– The encryption process does not rely on the key values only. Their combining form is also important. It has a flexible structure which is not previously established and depends on the user's choices,

– Permutation and substitution are combined within the same operation,

– The algorithm does not depend on the machine's precision neither in encryption nor in decryption processes. It uses very low precision in computing and provides high sensitivity.

The rest of the paper is organized as follows. The next section explains the projection process. It gives the underneath mathematical basis of the proposed projection. The third section explains how an image is encrypted. It gives the algorithm, its complexity, the structure of the used key, and explains how the reverse process of decryption is done. The fourth section gives an analysis about the key space values and its sensitivity. The fifth section is interested in the analysis of the resistance of the algorithm against some types of attacks. Finally, a conclusion is given summarizing the work and its extensions.

2 Projection process

Figure 1 shows a cut of the reference position from which an image is assumed to be viewed in its normal appearance, and another view's position rotated about the central axis. The latter position involves a rotated plane that is perpendicular to the direction of the viewer's position onto which the image is to be projected. Such position is defined by its inclination angle from the reference one.

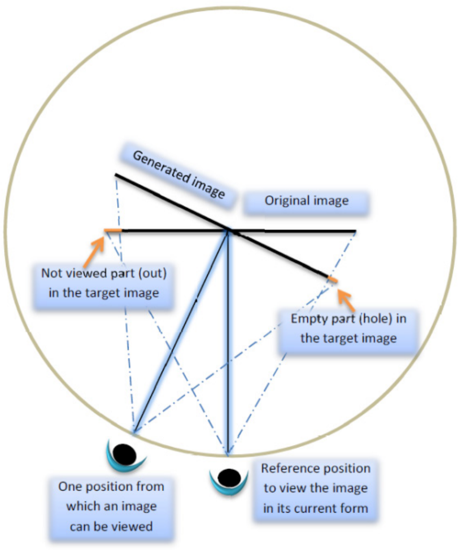

Generated image
Original image
Not viewed part (out) in the target image
Empty part (hole) in the target image
One position from which an image can be viewed
Reference position to view the image in its current form

Figure 1: Transformation of an original image into a new one when viewed from another position.

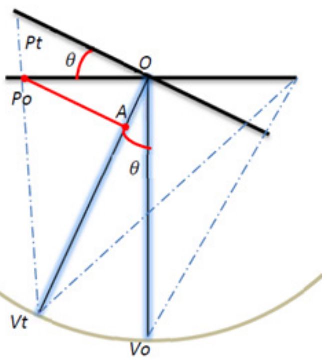


Figure 2: Projection of an original point P_o onto P_t in the new image.

The projection process is based on two assumptions. Firstly, the image is considered as three color planes in their low level data arrangement; i.e. the rows of each plane are vectors of bits where pixels are the groups of consecutive b bits depending on the image's gray level, and columns are the bits of the same order of image's columns of pixels. Secondly, each bit is assumed having a light ray that converges towards the viewer's position. Such ray crosses the rotated plane at a given point which is displaced from the bit's initial position. The new position is then used to paste the bit's value.

Figure 2 gives the schema of how a point P_o is projected at the position P_t when the viewer moves around the central axis of the image from the position V_o to the position V_t. Therefore, the achievement of the projection relies on the mathematical expression which governs the amount of OP_t.

Let the angle between the directions of the reference position and the new one be θ (figure 2). This angle is the same between the original image plane and the rotated one. Let now A be the perpendicular projection of P_o on OV_t. Using ΔOP_tV_t and ΔAOP_o, it is easy to see that $AP_o = OP_o.cos\theta$, and $OA = OP_o.sin\theta$. Using these expressions, and based on the truth of expression (1):

$$\frac{OP_t}{AP_o} = \frac{OV_t}{AV_t} \qquad (1)$$

the amount of OP_t can be computed for the half part of the image being on the same side of the viewer's position (2):

$$OP_t = \frac{OV_t.OP_o.cos\theta}{OV_t - OP_o.sin\theta} \qquad (2)$$

This theory is applied in either horizontal or vertical direction. Hence, expression (2) is used to project the bits of each half line or half column of the original image onto the same half line or half column of the new image. So, each bit in the original image at the position P_o is projected onto the position P_t. Such transformation needs two basic parameters to be known: the angle θ and the distance $OV_o = OV_t$. Practically, we take the value of the distance as being $d.lc$ where lc stands for the number of lines or columns, and d is a real such that $d \geq \frac{1}{2}$. The latter condition is important since it ensures that $OV_t - OP_o.sin\theta \neq 0$, because in this case we get $sin\theta = \frac{OV_t}{OP_o} = \frac{d.2OP_o}{OP_o} = 2d \geq 1$ which is impossible according to the previous condition. The case where $sin\theta = 1$ is meaningless since it is practically impossible. It gives $\theta = \frac{\pi}{2}$, making the projection of the whole image being one point and encryption process is reduced to inversion of bits' positions.

Note that the relationship (3) that links P_t and P_o on the opposite side of the viewer's position is so complicated, and consequently, has a negative impact on the algorithm complexity and time processing. This is why, the only expression used in this work on both sides of an image is (2), and since expression (3) is not used, it is given without demonstration.

$$OP_o = OP_t.cos\theta + \frac{OP_t.sin\theta}{cot\left(arctan\left(\frac{OP_t}{OV_t}\right) - \theta\right)} \qquad (3)$$

The projection of each line or column of the original image on a new one has some effects. If the angle θ is small, a set of collated bits on the border, called P_{out}, are projected outside the boundary of the new image (figure 3(a)). This creates stretched lines or columns with many spread undefined bits, since a reduced number of bits become spread over the whole line or column. If θ is big, it results in an undefined part in the projected image (figure 3(b)). Furthermore, due to the rounding of the estimated value of OP_t to its nearest integer, some bits are going to have the same projection positions, especially when an empty part is created for a big θ. In this case, only one bit is pasted at the shared position and the others are not. All those which are not pasted create the set called P_{rep}. The two sets, P_{out} and P_{rep}, are initially delayed from projection. As a result, the obtained image might contain several undefined bits called holes.

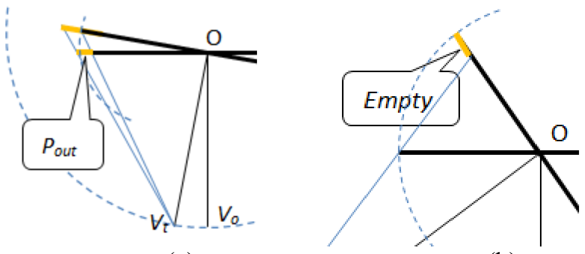


Figure 3. Effects of θ and d on the projection: (a) some bits are projected outside, (b) part of the border remains empty.

The value of θ_l which is the limit between those that create either P_{out} or an empty part is given when OP_t and OP_o become equal. In other words, if $\theta > \theta_l$ then $P_{out} = \emptyset$. So, θ_l is the angle when $OP_o = OP_t = \frac{lc}{2}$. Expressing this condition in (2) gives $lc = \frac{2d.lc^2.cos\theta_l}{2d.lc - lc.sin\theta_l}$, which results in the following equation:

$$2d.cos\theta_l + sin\theta_l = 2d \qquad (4)$$

The solution of (4) is the one of:

$$cos(\theta_l - \beta) = \frac{2d}{\sqrt{1+4d^2}} \qquad (5)$$

where $tan\beta = \frac{1}{2d}$. The solution to (5) is then:

$$\theta_l = arccos\left(\frac{2d}{\sqrt{1+4d^2}}\right) + arctan\left(\frac{1}{2d}\right) \qquad (6)$$

and

$$\lim_{d\to+\infty} \theta_l = 0 \qquad (7)$$

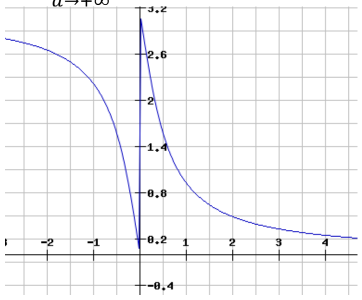


Figure 4: Graph of θ_l related to function (6).

The graph of θ_l (figure 4) shows that the bigger d is, i.e. the viewer is far from the image, the smaller θ_l is. In the graph of figure 4, the plot of the function where $d \geq \frac{1}{2}$ is the only part of our interest.

3 Image encryption

3.1 Cipher procedure

The first step of shuffling procedure is the projection of each line or column on the same line or column respectively of the new image. This goes through a vector calculation linking original to target bits' positions according to expression (2). For the lines, all bits are pasted in their new estimated positions. However, not all columns of bits are concerned with vertical projection. The bits of high and middle orders of each pixel are the only ones concerned. This leads to an exchange of only these bits between the pixels of the same column. This is so because if all bits of pixels are used, the projection has a tendency to glide a whole line of pixels the same way. The vertical projection becomes then a simple shift of the image. Moreover, the bits of high order are the most probable to make a significant change on pixels even though this depends on the content of each one.

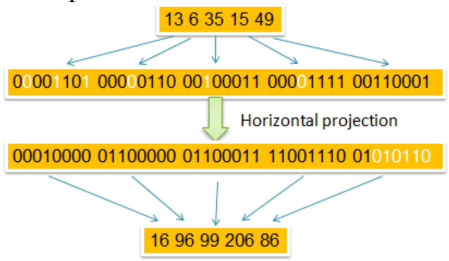


Figure 5: A sample of horizontal projection resulting in the bits shifted to the left (for this case) and some ones (white) reintroduced from the right.

The next step after image projection is to fill the created holes with the delayed bits, i.e. P_{out} and/or P_{rep}. To do so, each bit is reintroduced into the opposite half part from which it does not originate and in an opposite order of its stacking. In addition, P_{out} are reintroduced first in order to be stretched out in the image because they are initially contiguous. Then, P_{rep} are simply pasted into the remaining holes in a similar way of P_{out} (Figure 7).

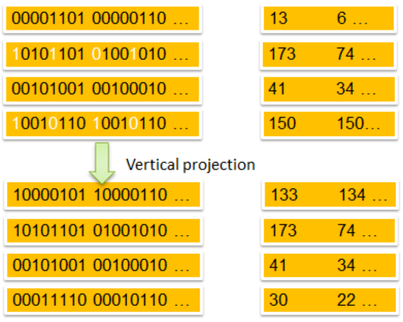


Figure 6: A sample of vertical projection resulting in the bits (3 and 7 of each pixel) shifted in vertical direction to the bottom and some ones (white) reintroduced from the top. The right column shows the corresponding decimal values.

To sample what has been explained, let us consider figures 5 and 6. When the transformation is applied in horizontal direction, some bits of each pixel either shift or migrate towards another one of the same line, and consequently, all pixels of each line change their values (figure 5). Likewise, when the projection is applied in vertical direction, the exchange of bits is done between the pixels' bits of one column (figure 6). In this case, only the bits of high and middle orders are projected as this is explained later.

The shuffling procedure of an image is, in general, based on repeating n times the cycle of new image projection and reinsertion of delayed bits. In fact, several repetitions of such process on the successively new obtained images are enough to get a complete cipher image. Furthermore, the algorithm combines several couples (θ, d) in several ways. It may apply the shuffling procedure alternatively in vertical and horizontal directions with different values, i.e. (θ_x, d_x) and (θ_y, d_y), at different steps with different number of each, and with different combinations for each color plane of the image.

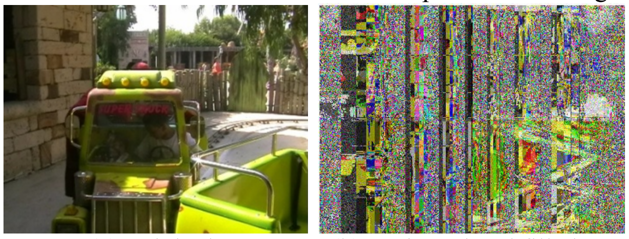


Figure 7: An original image (a) and its projected one (b) using $\theta = 1°$ and $d = 0.5$ for both horizontal and vertical directions.

3.2 Permutation and substitution of pixels

The power of the previously described process is that it results in both permutation and substitution of image's pixels (figures 5 and 6). In fact, as it is already explained, due to the projection process, the bits are displaced from their positions. At first glance, the consecutive pixel's bits are dragged together. This can be seen as pixels' movement resulting in a confusion procedure. From another point of view, pixels do not move really, they are the bits which do it. So, depending on θ, some bits of each pixel are going to move to another one. Thus, each pixel gets a new value; i.e. it is substituted or diffused. This is what happens mainly in vertical projection. Consequently, based on both points of view, the pixels' values can be considered changing when moving. This is what creates a simultaneous permutation and substitution of pixels within one operation without being discernible.

3.3 Key structure

In this work, the parameters θ and d serve as parameters of encryption key. They are used in couple at any step of shuffling. Therefore, different forms of the key may be used, and can be as complex as needed. For this, the algorithm does not depend on any structure. It can be adapted easily to anyone. Considering our experimentations, the adopted structure is as follows:

$$[nb_{iter}] \quad [v_i] \quad [\theta_{x_1},\dots,\theta_{x_{v_i}}][d_{x_1},\dots,d_{x_{v_i}}] \quad [h_i] \quad [\theta_{y_1},\dots,\theta_{y_{h_i}}][d_{y_1},\dots,d_{y_{h_i}}]$$

The used structure implies a global number of iterations (nb_{iter}) applied to local couples vertically (v_i) and horizontally (h_i). Each of which is of the form (θ_{axis_i}, d_{axis_i}), where $axis = x$ or y.

As an example of its use, figure 8 shows the encryption of the image of figure (7(a)) with two different keys. The one of figure (8(b)) is a special case where each color plane is scrambled apart with different key values.

As it is clear, the used key is compounded of many parts having different sizes. For each angle, 1 bit is used for the integer part and 10 bits for the decimal part. Since the distance is a multiple of the line or column's length, it uses 4 bits for the integer part and 8 bits for the decimal part. In sum, each couple (θ, d) uses a total of 25 bits together with two bits used for the number of local couples (θ, d) in a given direction. The total number of global iterations uses 6 bits allowing the maximum of 64 iterations which might be highly paid in term of execution time. The size of the key is then 6+25(v_i+h_i) bits. At least, this size is 56 bits.

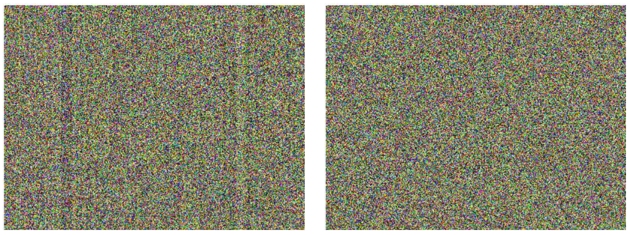

(a) $Key = K_1$ (b) $Key = K_1/K_1 + 1/K_1 + 2$

Figure 8: Scrambling of the image in figure 7(a). (a) is shuffled using $K_1 = [4,1,\frac{5\pi}{18},2.0,1,\frac{\pi}{3},1.6]$, but in (b) K_1 is used for only the color plane R. G and B color planes use K_1 where each couple (θ, d) is increased respectively by the values (1, 1) and (2, 2).

3.4 Algorithm complexity

The basis of the whole process is equation (2). Considering that $OV_o = OV_t = d.lc$, the expression becomes:

$$OP_t = \frac{d.lc.OP_o.cos\theta}{d.lc - OP_o.sin\theta} \quad (8)$$

In (8), two multiplications, one division, and one subtraction are uses. This is so, because $d.lc.cos\theta$ and $sin\theta$ are computed once at the beginning. So, taking $N \times N$-sized image, computing complexity of projection calculation for a key of (v_i+h_i) couples (θ, d) is $\frac{(v_i+Lh_i)(a+3b)}{2}N$, where a is the cost of a subtraction operation, b is the one of a multiplication or division operation, and L is the number of each pixel's bits. We get Lh_i because in horizontal direction the computing is related to bits, whereas in vertical one it is related to regular lines. The division by 2 is due to the consideration of (8) on only one half of the image.

In horizontal direction, the projection is applied on lines being N vectors. Thus, let p be the cost of changing one bit, the complexity is then LpN^2. In vertical direction, it is $2pN^2$ since only two bits of each pixel are concerned. For the whole algorithm, the complexity is then $\frac{(v_i+Lh_i)(a+3b)}{2}NL + h_ipLN^2 + v_i2pN^2$ and with an iteration time n, it becomes:

$$\frac{(v_i + Lh_i)(a + 3b)}{2}N + np(h_iL + 2v_i)N^2 \quad (9)$$

For vertical direction processed the same way of horizontal one, we get:

$$\frac{(v_i + h_i)(a + 3b)}{2}LN + np(h_i + v_i)LN^2 \quad (10)$$

In its lowest level of security, this algorithm uses a key compounded of two couples (θ, d), one for each direction. In this case, $h_i = v_i = 1$ which makes the complexity being $(a + 3b)\left(\frac{L+1}{2}\right)N + np(L + 2)N^2$. Compared to the ones of the three compared algorithms in [12], this complexity is lower than the lowest one of Baker map algorithm which is $2(a + b)N^2 + n(a + b)N^2$. As can be seen, $\left(\frac{L+1}{2}\right) \ll N$ which makes the other coefficients negligible. Even if all bits of each pixel are considered in vertical direction, the complexity $(a + 3b)LN + 2npLN^2$ remains the lowest.

3.5 Reconstruction of the original image

The retrieval of the original image is based on a reverse process of its encryption. Knowing the values of encryption key, the reconstruction of the original image follows exactly the reverse steps. In other words, the algorithm piles up the operations which are then executed from the last one to the first one having its steps reversed for the following points:

– lines - columns. For instance, if for a given step of the encryption process the lines were scrambled first then the reconstruction should begin by the columns first,

– d and θ. This is the case when the key form uses different values for these parameters at the same iteration. For instance, if (d_1, θ_1) and (d_2, θ_2) are two couples of values used in this order for image encryption, the order (d_2, θ_2)-(d_1, θ_1) should be used during the decryption,

Considering P_{out} and P_{rep}, they are first identified through the initially created holes in the cipher-image. The reverse operation consists then in copying the bits at the positions of the holes from the encrypted image into their initial positions in the reconstructed image. This is done with respect to the reverse order of their initial copy during image encryption. This is evident for P_{out}. For P_{rep}, however, the algorithm should differentiate between the first bits which were directly projected and those which were delayed. So, the latter are the only ones that are retrieved from the holes. In sum, these operations are executed whenever they were done during the encryption process for either P_{out} or P_{rep}.

4 Key analysis

4.1 Key space

The numbers of significant values of θ and d together with their combining form have direct effects on the possible values of a key. These last are in close relation to the sensitivity of encryption to the key values. In the following, a presentation of the key space according to the used structure in our experiments then to its general form is given.

According to the chosen structure of the key, one can see that the key subspace of the couple (θ, d) is $2^{11}.2^{12} = 2^{23}$. For a maximum number of local different couples being 3, this value becomes $(2^{23})^3 = 2^{69}$. Since this is applied in horizontal and vertical directions with different values, the total number is $2^{69}.2^{69} = 2^{138}$. This is for a key size of 156 bits. If different keys are used for each color plane, the total number is then $2^{138}.2^{138}.2^{138} = 2^{414}$. Additionally, if different keys are used at different iteration times n, this value increases to become $(2^{138})^n$ or $(2^{414})^n$.

According to the general form of the key, let the number of values of θ be θ_v and those of d be d_v. The space values of a couple (θ, d) is $\theta_v \times d_v$. For a number of different couples used in a combining form of the key, this becomes $(\theta_v \times d_v)^{(h_i+v_i)}$. If different keys are used for each color plane, we get $(\theta_v \times d_v)^{3(h_i+v_i)}$. Moreover, if different keys are used in different iterations, then the key space becomes $(\theta_v \times d_v)^{3n(h_i+v_i)}$ which is very big. Furthermore, if high level of security is not required, d_v may be very big.

As can be seen, the key space increases with h_i, v_i, and n. However, not all possible values of θ and d are used for secure encryption. The next section explains the raisons. Hence, the choice of h_i, v_i, and n depends on the required security and complexity. So, for high level of security and when some values of θ and d are avoided, the values of h_i, v_i, and n should increase, and vice versa. Note that all next experiments use $h_i = v_i = 1$ and $n \leq 4$ which provide the lowest security level of key structure in order to show the power of this algorithm.

4.2 Key sensitivity

In order to specify the significant values of a key that lead to a given level of security against differential attack, some experiments have been done. One of them is the use of ciphertext difference rate (Cdr) defined in [12, 17] and adapted to our case for color images as:

$$Cdr = \frac{\sum_c \left(Diff_c(Y, Y_1) + Diff_c(Y, Y_2)\right)}{6NM} \quad (11)$$

Whereas index c indicates one color plane R, G, or B and

$$Diff_c(A_c, B_c) = \sum_i \sum_j Difp_c\left(A_c(i,j), B_c(i,j)\right) \quad (12)$$

and

$$Difp_c\left(A_c(i,j), B_c(i,j)\right) = \begin{cases} 1 \text{ if } A_c(i,j) \neq B_c(i,j) \\ 0 \text{ if } A_c(i,j) = B_c(i,j) \end{cases} \quad (13)$$

Y is the cipher image of size $N \times M$ under a key K, Y_1 under the key $K + \Delta K$, and Y_2 under the key $K - \Delta K$.

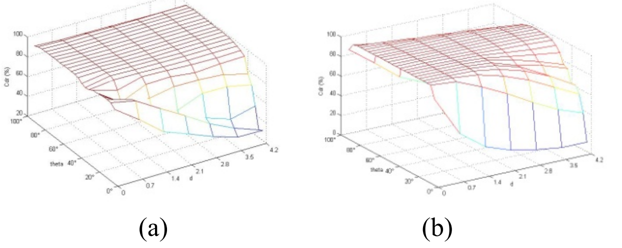


Figure 9: Cdr computing based on a slight difference of θ using 3 iterations and (a) $\Delta\theta = 0.001°$ (b) $\Delta d = 0.01$.

In our tests, we used $\Delta\theta = 0.001°$ for θ (figure 9(a)) and $\Delta d = 0.01$ for d (figure 9(b)) separately. In these experiments, the structure of the very low security of the key is used; i.e. each key has only one and the same couple (θ, d) for each direction and for all color planes. Figure 9(a) shows that for $\theta \in [20°..89°]$, the security of the proposed cryptosystem is acceptable against brute-force attacks since $Cdr \geq 90\%$. Figure 9(b) shows again this power when $d \in [0.5..3]$ for approximately the whole θ range. Note that these results are obtained for only three iterations. This is as good as Cat map algorithm in [12] which is the best of the three presented algorithms according to this test.

Consequently, for high level of security, it is advisable not to use all values of θ and d. In this case, $\theta \in [20°..89°]$, $d \in [0.5..3]$ and four iterations are the most advised. What is important in the algorithm is that for such high level of security, it does not require high cost. In addition, the key sensitivity is of high level since a difference of 1°/1000 for θ and 1/100 for d has considerable effects on the cipher image.

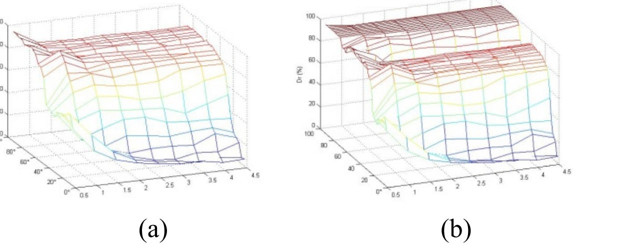
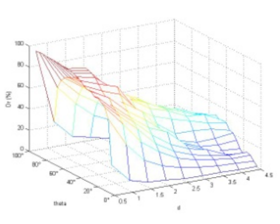


Figure 10: Dr computing based on $\Delta\theta = (10^{-3})°$ using (a) n=1 and (b) n=4.

The previous results are supported by other conducted experiments which aim to specify the tolerance interval $\pm\Delta\theta$ and $\pm\Delta d$ of decryption sensitivity. These experiments have been conducted according to two ways. The first one is subjective. It consists in considering the opinions of some people whether they are able or not to identify the content of the decrypted image using an erroneous key. This shows that the values of $(10^{-3})°$ for θ and 10^{-2} for d are the limits of image identification if some particular values are avoided. In other words, $\pm10^{-3}$ and $\pm10^{-2}$ are the tolerance intervals in order for the use of a false key does not decrypt the image.


(a)

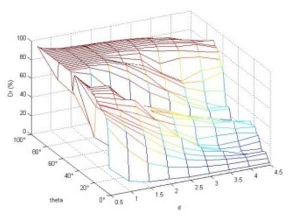

(b)

Figure 11: Dr computing based on $\Delta d = 10^{-2}$ using (a) n=1 and (b) n=4.

Based on these results, the second tests are based on the computing of the difference between the original image and the decrypted one. For this, we propose the calculation of the decryption rate (Dr) as:

$$Dr = \frac{\sum_c Diff_c(Y, Y')}{3NM} \tag{14}$$

The function $Diff_c$ is defined in (12), Y is the original image, and Y' is the decrypted image using the key $K' = K \pm \Delta K_i$, $(i = \theta, d)$. K is the encryption key and ΔK_i =(10^{-3} or 10^{-2}) is the slight change of only one occurrence of one parameter of K.

These tests use the form of the key which is of the very low security level as in Cdr tests. The graphs of figures (10) and (11) show the obtained results. The used iteration time is either 1 or 4. As can be seen in all cases, Dr increases with the number of iteration time n.

The graph of figure 10 is obtained for $\Delta\theta = (10^{-3})°$ and $\Delta d = 0$, and the one of figure 11 for $\Delta\theta = 0°$ and $\Delta d = 10^{-2}$. What is noticeable for the graph of figure 10 is that for $\theta \geq 20°$, Dr remains very acceptable even when d increases, whereas in figure 11, it falls off quickly for big values of d. This leads us to conclude again that the decryption process is more sensitive to θ than to d.

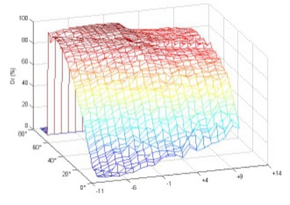
Throughout the conducted experiments, it can be concluded that small iterations (≤ 3) and small values of $\theta \in [1°..10°]$ are not advisable for high security if the used key is of low security level.

4.3 Effect of θ_l on encryption

To study the effect of θ_l on the encryption, Cdr and Dr are again considered. Since θ_l exists for each value of θ related to a specific distance d, we used to vary θ_l from 10° to 80°, compute the corresponding d, and evaluate Cdr and Dr with regard to all angles of the interval $\theta_{l_i} \pm j°$ $(i = 10°..80°,\ j = 1°..10°)$. For this purpose, expression (4) which links θ_l to d is used to determine each d_i related to a given θ_{l_i}. In this case, we get:

$$d = \frac{sin\theta_l}{2(1 - cos\theta_l)} \tag{15}$$

Through the graphs of figures 12 and 13, one can see that for good Dr and Cdr, the best values are obtained for $\theta < \theta_l$ and the quality decreases for $\theta > \theta_l$, even though those of Cdr are not so notable. In other words, the results of this cryptosystem are better when P_{out} are not empty and its efficiency decreases with the creation of an empty part but remains very acceptable. This is logic since P_{out} leads to more scrambling of bits than when an empty part is created. Moreover, the sensitivity is more important for $\theta < \theta_l$ than the one of $\theta > \theta_l$ except for some very small values of θ as has been already noticed. Note also that the graphs of figures 12 and 13 highlights again the sensitivity of the cryptosystem to θ than to d.


(a)

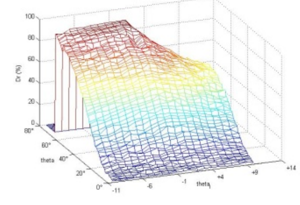
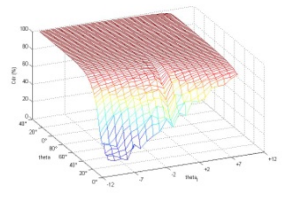

(b)

Figure 12: Dr computing for every $\theta_l \in [10°..80°]$ using 3 iterations based on (a) $\Delta\theta = (10^{-3})°$ and (b) $\Delta d = 10^{-2}$.


(a)

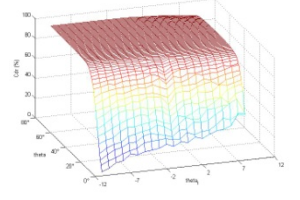

(b)

Figure 13: Cdr computing for every $\theta_l \in [10°..80°]$ using 3 iterations based on (a) $\Delta\theta = (10^{-3})°$ and (b) $\Delta d = 10^{-2}$.

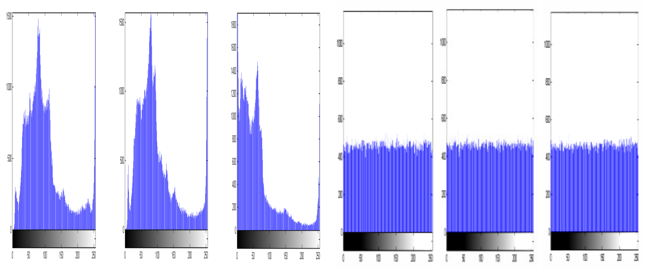

(a) red (b) green (c) blue (d) red (e) green (f) blue

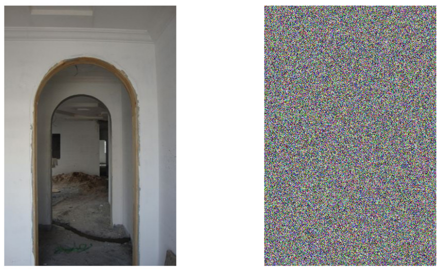
Figure 14: Colour histograms of the two previous images: (a), (b), and (c) of the original image (Figure 7(a)), and (d), (e), and (f) of its corresponding encrypted form (Figure 8(b)).

5 Resistance against other attacks

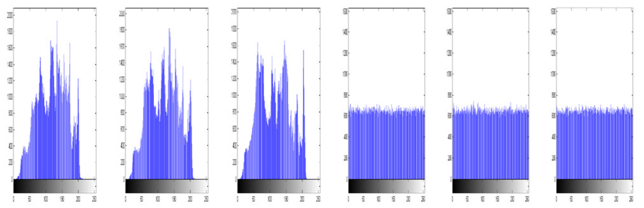
5.1 Statistical attack

Figure (8(b)) shows the result of the encryption process applied to the image of figure (7(a)), and their histograms are shown in figure 14. It is clear that the histograms of the two images, from statistical point of view, are significantly different. The ones of the encrypted image are flat and totally misleading, whereas those of the original image are curved.

Moreover, figures (15), (16), and (17) show other images which were used in our experiments, and on which this encryption algorithm is applied. These images have been chosen based on the different shapes their color histograms have. It is clear that the histograms of the original images and encrypted ones are totally different. Even though the ones of the three images are of different forms, those of encrypted images are completely flat.



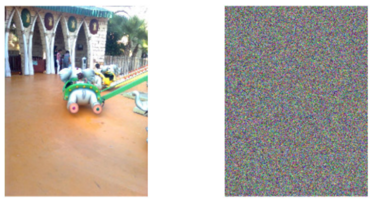
(a) Original (b) Encrypted



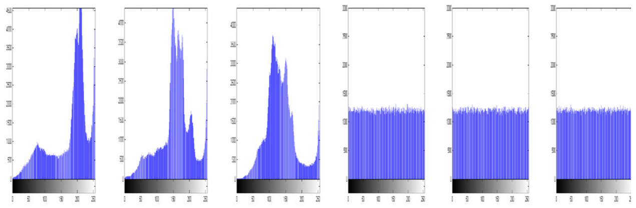
(c) red (d) green (e) blue (f) red (g) green (h) blue

Figure 15: Histograms of an original image (a) and its encrypted form (b) where the histograms of the former ((c), (d), and (e)) have several peaks concentrated in the middle.

5.2 Select plaintext attack



(*a*) Original (*b*) Encrypted

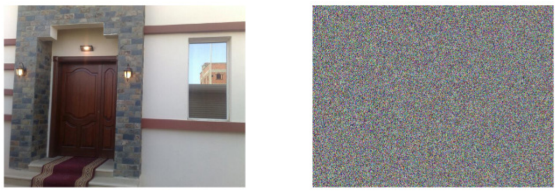


(c) red (d) green (e) blue (f) red (g) green (h) blue

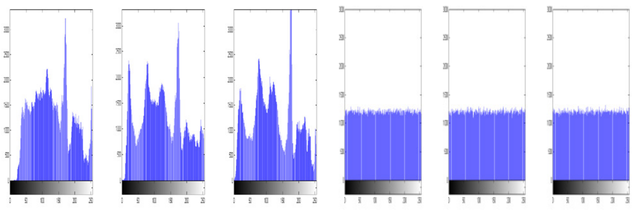
Figure 16: Histograms of an original image (a) and its encrypted form (b) where the histograms of the former ((a), (c), and (d)) have approximately one peak.

Another feature the algorithm should resist is select plaintext attack. It is commonly used to analyze plaintext sensitivity; i.e. the effect of the algorithm on parts with little differences. To test such feature, we use pixel change rate (Pcr) defined in [12] and adapted to our case as:

$$Pcr = \frac{\sum_c Diff_c(Y, Y'')}{3NM} \tag{16}$$



(a) Original (b) Encrypted



(c) red (d) green (e) blue (f) red (g) green (h) blue

Figure 17: Histograms of an original image (a) and its encrypted form (b) where the histograms of the former ((a), (c), and (d)) have many peaks spread out along the dynamic range.

In this case, Y and Y'' are encryption results using the same key of images I and $(I + \Delta I)$. ΔI is a slight change in one bit for instance; i.e. an image and its transformed one having one bit different are ciphered. The graph in figure 18 shows that this algorithm maintains Pcr at a stable level depending on the initial change. This is explained by the fact that initial bits of the original images are the same about $\frac{L-1}{L}$ for one pixel different. Since there are no significant differences, the two images evolve the same way during encryption. Even though Pcr does not reach 100%, it remains so important being $\geq 64\%$ in all cases.

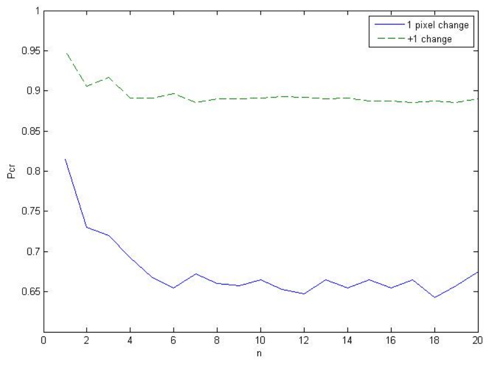


Figure 18: Pcr graph when a slight change happens to each pixel of an image.

6 Conclusion

A new algorithm of image encryption based on a geometrical transform is presented. The algorithm projects an original image on a plane perpendicular to the direction of a given view's position. It considers the image as rows and columns of bits. So, the projection of each bit is done on the position where its light ray intersects the perpendicular plane to the direction of the view. As a result, a cipher process takes place. This process is repeated several times to ensure a given level of security.

The strength of the algorithm depends mainly on the structure of the used key. The core of the algorithm is based on two parameters: the angle of view and the distance of the viewer from the image. They are used in couples, and combined in many ways so that the combination itself becomes a key. Hence, the power of this algorithm is that the combinations depend on the genuineness of the user.

The efficiency of this algorithm is demonstrated through several tests. Its key sensitivity from many points of view is about $(10^{-3})°$ for the angle and 10^{-2} for the distance. Moreover, its resistance to many common attacks is proved through many graphs of the very low level of security provided by both key structure and values.

Further extensions for this work may be made in different directions. One of them is the study of the algorithm resistance to different attacks apart from the security considerations. For example, to what extent an encrypted image may resist to JPEG compression in order to reconstruct the original one since the initial bits are not explicitly changed. The structure of the key is also important. One possible improvement is to make a formal study of it. Another important study is about the general form of the projection. It may be part of the key.

References

[1] B. Nini and C. Melloul (2011), Pixel Permutation of a Color Image Based on a Projection from a Rotated View. *JDCTA: International Journal of Digital Content Technology and its Applications*, Vol. 5, N° 4, pp. 302-312.

[2] Chengqing Li, Kwok-Tung Lo (2009). Security analysis of a binary image permutation scheme based on Logistic map. http://arxiv.org/pdf /0912.1918

[3] Chengqing Li (2008). On the Security of a Class of Image Encryption Scheme. *Int. Symposium on Circuits and Systems,* IEEE, pp. 3290-3293.

[4] Ercan Solak, Cahit Cokal and Olcay Taner Yildiz (2010). Cryptanalysis of fridrich's chaotic image encryption. *International Journal of Bifurcation and Chaos*, *DOI: 10.1142/S0218127410026563*, Vol. 20, n° 5, pp. 1405–1413.

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

[6] G. Ye (2009). Image scrambling encryption algorithm of pixel bit based on chaos map. *Pattern Recognition Letters, DOI: 10.1016/j. patrec. 11.008*.

[7] 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.

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

[9] John Vreugdenhil, Kane Iverson, and Raj Katti (2009). Image Encyption Using Dynamic Shuffling and XORing Processes. *IEEE International Symposium on Circuits and Systems*, ISCAS, pp. 734–737.

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

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

[12] Shiguo Lian, Jinsheng Sun, Zhiquan Wang (2005). Security Analysis of A Chaos-based Image Encryption Algorithm. *Physica A: Statistical and Theoretical Physics, Elsevier*, 351(2-4): pp. 645-661.

[13] Shujun Li, Chengqing Li, Guanrong Chen, and Kwok-Tung Lo (2008). Cryptanalysis of the RCES/RSES image encryption scheme. *The Journal of Systems and Software*, *DOI:10.1016/j.jss.2007.07.037*, pp. 1130–1143.

[14] Trenkler M. (2005). An Algorithm for making Magic Cubes, *The Π ME Journal*, vol. 12, n°2, pp. 105-106.

[15] 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. *International Conference on Computer Science and Software Engineering*. IEEE, Los Alamitos.

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

[17] Shiguo Lian, Jinsheng Sun, Zhiquan Wang (2005). A Block Cipher Based on a Suitable Use of the Chaotic Standard Map. *International Journal of Chaos, Solitons and Fractals, Elsevier*, 26(1): 117-129.

[18] Yaobin Mao, Guanrong Chen and Shiguo Lian (2004). A Novel Fast Image Encryption Scheme Based on the 3D Chaotic Baker Map, *International Journal of Bifurcation and Chaos, World Scientific Publishing*, vol. 14, no. 10, pp. 3613-3624.

[19] Komal D Patel, Sonal Belani (2011). Image encryption using different techniques: A review. *International Journal of Emerging Technology and Advanced Engeneering,* vol. 1 no. 1, pp. 30-34.