DOI: http://dx.doi.org/10.26483/ijarcs.v8i7.4179

ISSN No. 0976-5697

Volume 8, No. 7, July – August 2017

International Journal of Advanced Research in Computer Science

RESEARCH PAPER

Available Online at www.ijarcs.info

QUANTUM IMAGE SCRAMBLING HAVING XOR USING 2D MEDIAN FILTERING

Dr.Rafiq Ahmad Khan
Assistant Professor GDC Bemina,
Kashmir,India

Mohd Iqbal Sheikh
PhD Scholar Mewar university
Chittorgarh, India

Abstract: With the rapid development of multimedia technology, the image scrambling for information hiding is severe in today's world. But, in quantum image processing field, the study on image scrambling is still few. Several quantum image scrambling schemes are in circle but, lot of it is yet to be performed. This paper presents the implementation of XOR quantum dot gate using bitwise operation to scramble an image metric. While the XOR operation has only half chance of outputting false or true (0, 1). XOR by scrambling an image so that image can be hidden immensely to avoid third party intervention. We use a non –linear method for removing the noise in this paper. . The medianfilterwas once the most popular nonlinear filter for removing impulse noise because of its good de -noising power and computational efficiency. Here we use 2D median filter

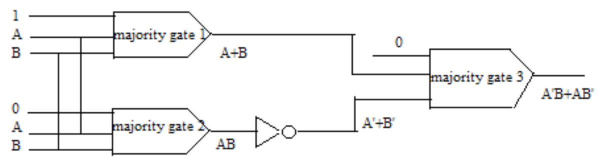
Keywords:Image scrambling Quantum image processing median**.**

INTRODUCTION

Quantum image processing is attracting more and more attention in recent years, from quantum image representation [1–3], quantum image operation [4–7] to quantum image encryption [8–10].Image scrambling [11, 12] is a basic work of image encryption or information hiding [13]. The image after scrambling removes the correlation of image pixels space, which can make the watermark lose the original information, and then, the watermark information is tucked into the carrier. Thus, even if an attacker extracted carriers from the image, he is almost unable to obtain the original image information in any case. Therefore, scrambling processing for the watermark or information hiding is fairly indispensable in a large sense. The scrambling algorithm mainly includes twocategories. Image bit-plane refers to a series of two-value image planes. To begin with, the pixel Values in the image are represented by its corresponding binary values, and then, every single bit of all the pixels will form a two-value image, it is called bit-plane. To bespecific, if the image gray value range is [0, 255].

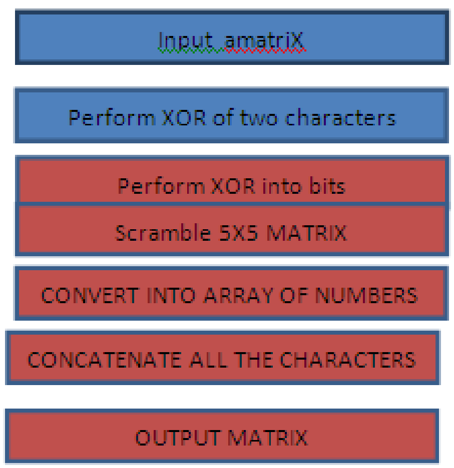
Two-input XOR (exclusive OR) also known as exclusive disjunction is a logical function which gives a highOutput only if any one of the two inputs but not both are high. The circuit diagram and the layout of XOR gate is shown in Fig 5(a) and Fig 5(b). The third input line of majority gate 1 is made high and that of majority gate 2 is made low. The output of majority gate 2 is fed into an inverter. Finally, the output from the majority gate 1 and that of the inverter is fed into majority gate 3 whose third input line is made 0. The output of majority gate 3 is the XORfunction. Impulse noise in an image is present due to bit errors in transmission or introduced during the signal acquisition stage. This noise is caused by malfunctioning pixels in camera sensors, faulty memory locations in hardware, transmission in noisy channel and external disturbance such as atmospheric disturbance [17].

Filters are designed as specific blocks and are used as masks for convolution operations. Basically two methods are used to remove the noise named as linear and Non-linear, and we use a non –linear method for removing the noise in this paper. . The medianfilterwas once the most popular nonlinear filter for removing impulse noise because of its good denoising power and computational efficiency. Here we use 2D median filter



(a): Circuit Diagram

WORKING: FLOWCHART:



Bitwise XOR operation to scramble two character matrices by generating a truth table .I need to perform the operation for four characters where each of them have a bit representation as follow

: XOR
A = 00
G = 01

C = 10
T = 11

I need to create a table thats two characters together which g ives the values for all combinations of
ing pairs of characters in the following way.

XOR	A	G	C	T
A	A	G	C	T
G	G	A	T	C
C	C	T	A	G
T	T	C	G	A

To obtain the output, you need to convert each character into its bit representation, the bits, then use the
result and convert it back to example, consulting the third row and second column of the table, by XORing C and G :

C = 10
C = 10
G = 01
C XOR G = 10 XOR 01 = 11 --> T

I would ultimately like to apply this rule to scrambling chara cters in a 5 x 5 matrix. As an example:

```
A =   'GATT'  'AACT'  'ACAC'  'TTGA'  'GGCT'
      'GCAC'  'TCAT'  'GTTC'  'GCCT'  'TTTA'
      'AACG'  'GTTA'  'ACGT'  'CGTC'  'TGGA'
      'CTAC'  'AAAA'  'GGGC'  'CCCT'  'TCGT'
      'GTGT'  'GCGG'  'GTTT'  'TTGC'  'ATTA'
B =   'ATAC'  'AAAT'  'AGCT'  'AAGC'  'AAGT'
      'TAGG'  'AAGT'  'ATGA'  'AAAG'  'AAGA'
      'TAGC'  'CAGT'  'AGAT'  'GAAG'  'TCGA'
      'GCTA'  'TTAC'  'GCCA'  'CCCC'  'TTTC'
      'CCAA'  'AGGA'  'GCAG'  'CAGC'  'TAAA'
```

I would like to generate a matri such that each element of A gets XORed with its corresponding element in B .C A XOR B.
For example, considering the first row and first column:
A{1,1} XOR B{1,1} = GATT XOR ATAC = GTTG
First, let's define the function
that takes two 4-character strings and bothstrings correspon ding to that table that you have.
, let's set up alookup table where a unique two-bit string corr esponds to a letter. We will also need the
lookup table using a class where given a letter, we produce a two-bit string. Wewant to convert each letter into its two b it representation, and we need the inverse lookup to-
do this. After, we XOR the bits individually, then use the forward lookup table to get back to wherewe started. As such:

```
function [out] = letterXOR(A,B)
   codebook = containers.Map({'00','11','10','01'},{'A','T','G','C'}); %// Lookup
   invCodebook = containers.Map({'A','T','G','C'},{'00','11','10','01'}); %// Inv-lookup
 lettersA = arrayfun(@(x) x, A, 'uni', 0); %// Split up each letter into a cell
   lettersB = arrayfun(@(x) x, B, 'uni', 0);
   valuesA = values(invCodebook, lettersA); %// Obtain the binary bit strings
   valuesB = values(invCodebook, lettersB);
   %// Convert each into a matrix
   valuesAMatrix = cellfun(@(x) double(x) - 48, valuesA, 'uni', 0);
ValuesBMatrix = cellfun (@(x) double(x) - 48, valuesB, 'uni', 0);
   % XOR the bits now
   XORedBits = arrayfun(@(x) bitxor(valuesAMatrix{x}, valuesBMatrix{x}), 1:numel(A),
'uni', 0);
   %// Convert each bit pair into a string
   XORedString = cellfun(@(x) char(x + 48), XORedBits, 'uni', 0);
   %// Access lookup, then concatenate as a string
   out = cellfun(@(x) codebook(x), XORedString);
```

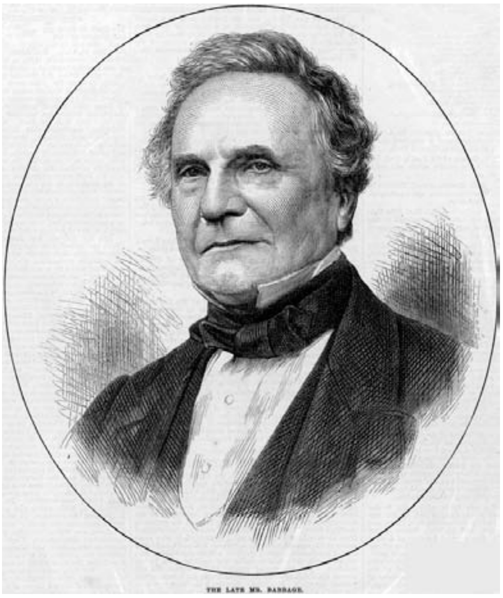
Let's go through the above code slowly. The inputs letterXOR are expected to be character array of letters that are composed of , , A,T,G,and C and . We first define the forward and reverse lookups. We then split up each character of the input strings A andB into a cell array of individual characters, as looking up multiple keys in your codebook requires it to be this way.
We then figure out what the bits are for each character in each string. These bits are actually strings, and so what we need to do is convert each string of bits into an array of numbers. We simply cast the string to double and subtract by 48, which is the ASCII code for 0 . By convertingto, you'll either get 48 or 49, which is why we need to subtract with 48.
As such, each pair of bits is converted into an
array of bits. We then take each 1x2 of bits between A and Bbitxor use
to xor the bits. The outputs at this point are still 1 x 2 after t his, we concatenate all of thecharacters together to make the final string for the output.Make sure you save the above in a function called. Once we have this, we now simply have to use one call that will XOR each four element string in your cell array andwe then output our final matrix. We will use to do that, and the input into will be a
matrix that is column major defined. We do this as MATLA B can access elements Ina 2D arrayUsing a single value. Thi s value is the column major index of the element in the matri x. We define a Vector that goes from 1 to 25, then use to get this into the right 2D form.
The reason why we need to do this is because we want to ma ke sure that the output matrix
(Which is
in your example) is structured in the same way. As such:

```
ind = reshape(1:25, 5, 5); %// Define column major indices
C  = arrayfun(@(x) letterXOR(A{x},B{x}), ind, 'uni', 0); %// Get our output matrix
```

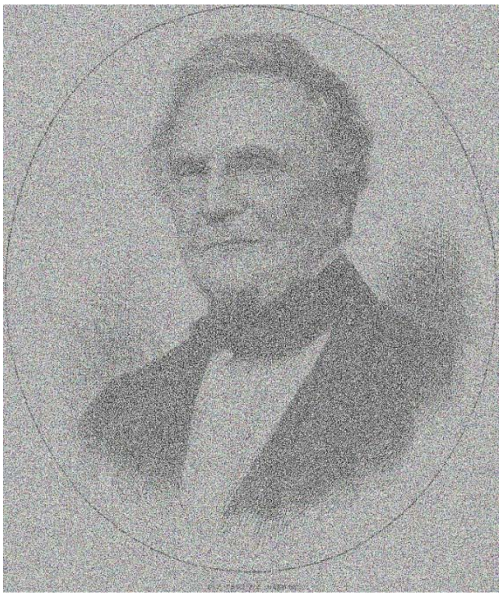
Our final output is:

```
C =
'GTTG'  'AACA'  'ATCG'  'TTAC'  'GGTA'
'CCGT'  'TCGA'  'GACC'  'GCCC'  'TTCA'
'TATT'  'TTCT'  'ATGA'  'TGTT'  'ATAA'
'TGTC'  'TTAC'  'ATTC'  'AAAG'  'AGCG'
'TGGT'  'GTAG'  'AGTC'  'GTAA'  'TTTA'
```

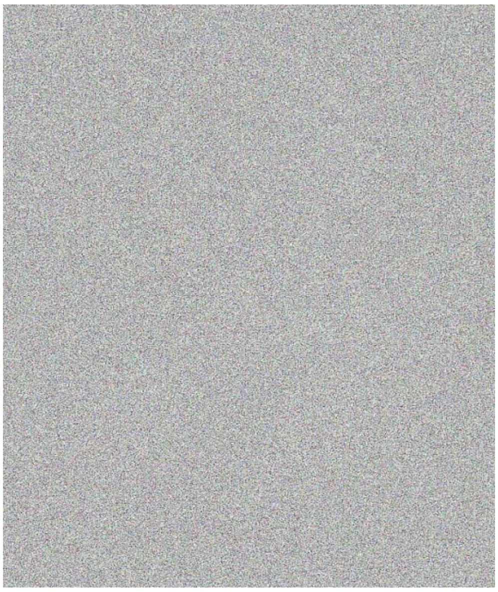

We use XOR, if we used AND, OR orXOR with the one-time and it's extremely important to understand that ANDhas a 75% chance of outputting 0 and a 25%chance of outputting a 1. While OR has a 25%chance of outputting 0 and 75% chance ofoutputting 1. While the XOR operation has a50% chance of outputting 0 or 1.XOR by encrypting an image. Here is adigital image of Charles Babbage:Let's look at a visual exampleto see thedifferent scrambling effects of AND vs. OR vsXOR by encrypting an image ... Here is digital image of Charles Babbage:



OR GATE USED



AND GATE USED



Xor used in this image contains no information about the original image. If we didn't provide the shiftsequence it would be impossible for you toreverse it back to the original image. You couldtry every possible sequence, but that wouldresult in every possible image! How could you Know it was Babbage? It's equally likely to be picture of you or anything else another thing to note about XOR versus ANDor OR is that it is reversible.

The truth table for XOR is:

0 0 | 0
0 1 | 1
1 0 | 1
1 1 | 0

So we know whenever we have 0 as the padbit, we can leave the bit as it is when Decrypting. When we have 1 as the pad bit, we flip the bit to get the decrypted bit.

The procedural step for the whole working of 2D median filtering for isas: a:

Consider a matrix A = $\begin{pmatrix} A & G & C \\ G & A & T \\ C & T & A \end{pmatrix}$

b: Now pad the matrix with zero on all the sides.

A= $\begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & A & G & C & 0 \\ 0 & 0 & G & A & T & 0 \\ 0 & 0 & C & T & A & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$

c: Consider a window of size 3 x 3.The window can be of any size. Starting from matrix A (1,1), place the window.

Window = $\begin{pmatrix} 0 & 0 & 0 \\ 0 & A & G \\ 0 & G & A \end{pmatrix}$

d: Window= $\begin{pmatrix} 0 & 0 & 0 \\ 0 & A & G \\ 0 & C & T \end{pmatrix}$

The value to be changed is the middle element [Value of 0 at(2,2)]

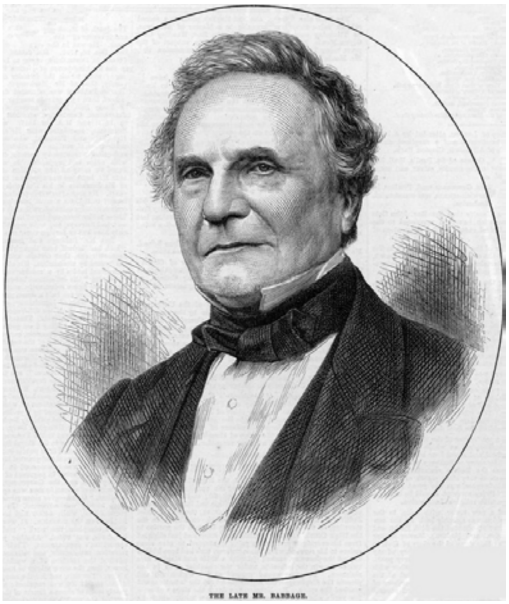
e: Sort the window matrix $\begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & T \\ C & C & G \end{pmatrix}$

f: After sorting the output matrix is placed with a value of 0 at (2,2) pixel position. The value of the output pixel is found using the median of the neighborhood pixels.

g: This procedure is repeated for all the values in the input matrix by sliding the window to next position i.e. A(1,2),and so on

h: The output matrix is= $\begin{pmatrix} 0 & T & 0 \\ A & C & A \\ 0 & A & 0 \end{pmatrix}$

The procedural steps we follow above are applied only when we revert the image back after decrypting using bitwise sequence shifting ,so that the image can be knowledgeable to the destination, then the image will be make free from the noise and the result is in the fig.



CONCULUSION

While the XOR operation has only half chance of outputting false or true (0, 1).XOR by scrambling an image so that image can be hidden immensely to avoid third party intervention. We use XOR, if we used AND, OR, XOR with the one-time and it's extremely important to understand that AND has a 75% chance of outputting 0 and a 25% chance of outputting a 1. While OR has a 25% chance of outputting 0 and 75% chance of outputting 1. While the XOR operation has a50% chance of Outputting 0 or 1.XOR by encrypting an image. The median filter was once the most popular nonlinear filter for removing imp ulse noise because of its good denoising power and computational efficiency.

REEFERENCES

1. Le, P.Q., Dong, F., Hirota, K.: flexible representation of quantum images for polynomial preparation, image compression, and processing operations. Quantum Inf. Process. 10(1), 63–84 (2011)
2. Zhang, Y., Lu, K., Gao, Y., Wang, M.: NEQR: a novel enhanced quantum representation of digital images. Quantum Inf. Process. 12(8), 2833–2860 (2013)
3. Li, H.S., Zhu, Q., Zhou, R.G., Song, L., Yang, X.J.: Multi-dimensional color image storage and retrieval for a normal arbitrary quantum superposition state. Quantum Inf. Process. 13(4), 991–1011 (2014)
4. Barenco, A., Bennett, C.H., Cleve, R., DiVincenzo, D.P., Margolus, N., Shor, P., Weinfurter, H.: Elementary gates for quantum computation. Phys. Rev. A At. Mol. Opt. Phys. 52(5), 3457 (1995)
5. Le, P.Q., Iliyasu, A.M., Dong, F., Hirota, K.: Efficient color transformations on quantum images. JACIII 15(6), 698–706 (2011)
6. Pang, C.Y., Zhou, R.G., Ding, C.B., Hu, B.Q.: Quantum search algorithm for set operation. Quantum Inf. Process. 12(1), 481–492 (2013)
7. Fijany, A., Williams, C.: Quantum wavelet transform: fast algorithm and complete circuits (1998). arXiv:quant-ph/9809004
8. Zhang, W.W., Gao, F., Liu, B., Wen, Q.Y., Chen, H.: A watermark strategy for quantum images based on quantum fourier transform. Quantum Inf. Process. 12(2), 793–803 (2013)
9. Iliyasu, A.M., Le, P.Q., Dong, F., Hirota, K.: Watermarking and authentication of quantum images based on restricted geometric transformations. Inf. Sci. 186(1), 126–149 (2012)
10. Zhou, R.G., Wu, Q., Zhang, M.Q., Shen, C.Y.: Quantum image encryption and decryption algorithms based on quantum image geometric transformations. Int. J. Theor. Phys. 52(6), 1802–1817 (2013)
11. Ye, G.: Image scrambling encryption algorithm of pixel bit based on chaos map. Pattern Recognit. Lett. 31(5), 347–354 (2010)
12. Dalhoum, A.L.A., Mahafzah, B.A., Awwad, A.A., Aldhamari, I., Ortega, A., Alfonseca, M.: Digital image scrambling using 2D cellular automata. IEEE Multimed. 4, 28–36 (2012)
13. Gunjal, B.L., Manthalkar, R.R.: Discrete wavelet transform based strongly robust watermarking scheme for information hiding in digital images. In: IEEE 3rd International Conference on Emerging Trends in Engineering and Technology (ICETET), pp. 124–129 (2010)
14. Monika Raghav, and SahilRaheja,"Image Denoising Techniques,"vol, 3, pp.5637-5641, issue5, May 2014.

15 .Kanika Gupta et al," Image denoising Techniques- a review paper,"vol.02, issue4, pp6-9March 2013.
16. RohitVarma,andDr.JahidAli,"A Comparative study of various types of noises and efficient noise removal techniques,"vol.3,issue 10,pp.617-622 oct.2013.
17. Srinivasan KS, Ebenezer D. A new fast and efficient decision-based algorithm for removal of high-density impulse noises. IEEE Signal Process Lett2007; 14:189–92

