Digital Images Encryption in Spatial Domain Based on Singular Value Decomposition and Cellular Automata

Ahmad Pahlavan Tafti
PhD Student, Department of Computer Science
University of Wisconsin Milwaukee
ahmad.pahlavantafti@ieee.org

Reyhaneh Maarefdoust
Sama technical and vocational training college, Islamic Azad University, Mashhad Branch,
Mashhad, Iran.
Maarefdost@yahoo.com

***Abstract*— Protection of digital images from unauthorized access is the main purpose of this paper. A reliable approach to encrypt a digital image in spatial domain is presented here. Our algorithm is based on the singular value decomposition and one dimensional cellular automata. First, we calculate the singular value decomposition (SVD) of the original image in which the features of the image are extracted and then pushed them into the one dimensional cellular automata to generate the robust secret key for the image authentication. SVD is used as a strong mathematical tool to decompose a digital image into three orthogonal matrices and create features that are rotation invariant.**

We applied our proposed model on one hundred number of JPEG RGB images of size 800 × 800. The experimental results have illustrated the robustness, visual quality and reliability of our proposed algorithm.

Keywords- Digital Images Encryption; Spatial Domain Encryption; Cellular Automata, SVD.

I. INTRODUCTION

Digital information like digital images and multimedia contents are widely used in many aspects such as meteorology, astronomy, radiology, robotics and surveillance systems. Validation and authentication of digital images are very important challenges for storing, retrieving and also transmitting of them.

Multimedia encryption has become the subject of very exhaustive research as its potential to transfer of information more securely. The encryption algorithms which have developed for text data are not suitable for multimedia data [1].

There are two main ways for digital images encryption. These are spatial domain and frequency domain encryption [2]. Spatial domain encryption is very simple where the frequency encryption is more complicated and reliable [2]. There are two level for digital images encryption; high-level and low-level. In the high-level encryption the content of the digital image is completely disordered and the original image is invisible. In low-level encryption, the content of the digital image is understandable and visible [3].

In this paper we focused on the spatial domain and low-level encryption methods. Our proposed model is based on SVD (Singular Value Decomposition) and one dimensional cellular automata. We use the singular value decomposition to extract the features of the original image (Singular Values and Singular Vectors) to push them into the cellular automata to create a secret key. This key is so much related to the digital image that any small change in the content of digital image will definitely change the key value without any exception.

The rest of this paper is arranged as follows. In section 2 we describe one dimensional cellular automatas and their rules. Section 3 introduces the concepts of SVD and its uses. Section 4 describes the system design and section 5 focuses on experimental results. Conclusions presents in section 6.

II. CELLULAR AUTOMATA

The history of cellular automata dates back to the 1940s with Stanislaw Marcin Ulam. This polish mathematician was interested in the evolution of graphic constructions generated by simple rules [4]. The base of his construction was a two-dimensional space divided into "cells", a sort of grid. Each of these cells could have two states: ON or OFF [5]. Cellular automata is a discrete dynamic model in space and time [5]. All of the cells arrange in the regular form and have a finite number of states. The states are updated with a local rule. Figure 1 shows a simple two state and one dimensional cellular automata with one line of cells. A specific cell can be either be on (value = 1= red) or off (value = 0= green). The closest cells to cell X are those to immediate left and right, moving along the lines connecting the nodes. The state of X at the time t + 1 will be determined by the states of the cells within its neighborhood at the time t. [6].

http://sites.google.com/site/ijcsis/
ISSN 1947-5500

Figure 1. One dimensional cellular automata with one neighborhood for cell X

We can set a local rule for each cellular automata. For example, we can estimate the value of cell X in time t+1 with the following rule [6]:

Cell[X] t+1 = Cell[X-1] t (OR) Cell[X+1] t

Assume that the input sequence is 01110 and we want to use the above rule for our cellular automata, then the output sequence will be 11111. Table 1 shows the output of this cellular automata.

TABLE 1. AN EXAMPLE OF CELLULAR AUTOMATA AND ITS RULE

Cell Number	**0**	**1**	**2**	**3**	**4**
Input Sequence (time t)	0	1	1	1	0
Cellular Automata Rule	$Cell[X]_{t+1} = Cell[X-1]_t$ (OR) $Cell[X+1]_t$				
Output Sequence (time t+1)	1	1	1	1	1

We use one dimensional cellular automata with XOR local rule to create a secret key which we want to embed this key into the spatial domain of a digital image. The input sequence in our proposed model is the array list of the sum and mean of eigenvalues and eigenvectors.

III. SINGULAR VALUE DECOMPOSITION

The basic theory of the SVD is reviewed in this section to show its power and ability to decompose any square or non-square digital image matrix into three orthogonal matrices that contain the useful features of the image. SVD can help us to select the dominant features in a digital image [7]. The SVD can decompose any real or complex n × p matrix into product of three matrices, an orthogonal matrix U, a diagonal matrix S, and the transpose of an orthogonal matrix V as (1):

$$A_{n\times p} = U_{n\times n}\, S_{n\times p}\, V^T_{pxp} \qquad (1)$$

Where U and V are Orthogonal Matrices ;i.e.

$$U^T U = UU^T = I_{n\times n} \qquad (2)$$

and

$$V^T V = VV^T = I_{pxp} \qquad (3)$$

Where the columns of U are called the Left Singular Vectors (Orthogonal Eigenvectors of AA^T), S (the same dimensions as A) a diagonal matrix that has the Singular Values (the Square roots of the Eigen values of AA^T or A^TA), and the columns of V called the Right Singular Vectors (Rows of V^T, Orthogonal Eigenvectors of ATA). The SVD represents an expansion of the original data in a coordinate system where the covariance matrix is diagonal [8].

To calculate the SVD of the matrix A we can either apply the Golub-Reinsch Algorithm that use a finite sequences of the Householder Transformation or directly find the Eigen Values and Eigen Vectors of AA^T and A^TA. The eigenvectors of A^TA make up the columns of V, the eigenvectors of AA^T make up the columns of U. where the singular values of S are the square roots of eigenvalues calculated from AA^T or A^TA. The singular values are the diagonal entries of the S matrix and are arranged in descending order. The singular values are always real numbers [9]. If the matrix A is a real matrix, then U and V are also real. Let us calculate the SVD of a 3×2 matrix A using the Eigen analysis of A^TA and AA^T.

$$A = \begin{bmatrix} 0 & 1 \\ 1 & 0 \\ 1 & 1 \end{bmatrix} \qquad (4)$$

First, we calculate $B=A^TA$ and apply the Eigen Analysis formula to get the Eigenvalues and Eigenvectors of B as (9):

$$B\,X = \lambda\, X \qquad (B - \lambda\, I)\, X = 0 \qquad (5)$$

$$\text{We have } B = \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix} \qquad (6)$$

Thus, the Eigenvalues and normalized Eigenvectors are:

$$\lambda_1 = 3,\ V_1 = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ 1 \end{bmatrix} \qquad (7)$$

$$\lambda_2 = 1,\ V_2 = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ -1 \end{bmatrix} \qquad (8)$$

The Singular values can be calculated as:

$$\sigma_1 = \|AV_1\|_2 = \sqrt{\lambda_1} = \sqrt{3}; \qquad (9)$$

$$\sigma_2 = \|AV_2\|_2 = \sqrt{\lambda_2} = 1 \qquad (10)$$

Thus we can immediately calculate u1, u2.

$$u_1 = \frac{1}{\sigma_1} AV_1 = \frac{1}{\sqrt{6}} \begin{bmatrix} 1 \\ 1 \\ 2 \end{bmatrix} \qquad (11)$$

$$u_2 = \frac{1}{\sigma_2} AV_2 = \frac{1}{\sqrt{2}} \begin{bmatrix} -1 \\ 1 \\ 0 \end{bmatrix} \qquad (12)$$

u3 must be selected such that to be orthogonal to both u1, u2. Thus, it can be written as:

$$u_3 = \frac{1}{\sqrt{3}} \begin{bmatrix} 1 \\ 1 \\ -1 \end{bmatrix} \qquad (13)$$

Therefore, A can be decomposed as products of three matrices:

$$\begin{bmatrix} 1/\sqrt{6} & -1/\sqrt{2} & 1/\sqrt{3} \\ 1/\sqrt{6} & 1/\sqrt{2} & 1/\sqrt{3} \\ 2/\sqrt{6} & 0 & -1/\sqrt{3} \end{bmatrix} \begin{bmatrix} \sqrt{3} & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} 1/\sqrt{2} & 1/\sqrt{2} \\ 1/\sqrt{2} & -1/\sqrt{2} \end{bmatrix} \qquad (14)$$

IV. PROPOSED MODEL

The main idea of our proposed algorithm is to create a robust secret key and embed it in the LSB of a specific layer of the original image, to encrypt it. Our proposed method is

http://sites.google.com/site/ijcsis/
ISSN 1947-5500

based on spatial domain and low-level encryption approaches in which some data or secret key is embedded into the spatial domain of the original image for the authentication. We have implemented our algorithm on a set of one hundred images and calculated the Eigenvalues and Eigenvectors of $B= A^TA$. Then derived the Singular Values and Right and Left Singular Vectors of the original image based on the equations (7-13) and pushed the SVD features into one dimensional cellular automata to generate the secret key. First, we achieve the Singular Values and the Right and Left Singular Vectors of the Red matrix (Red layer of the RGB image) derived from input image and perform the same task for the Green matrix. Then we use one dimensional cellular automata with a particular rule to create the secret key based on those values. Next, we embed the bit sequence of the secret key into the LSB of the particular pixels of the Blue layer (Blue Matrix) in the original image.

Our proposed algorithm performs on a RGB JPEG image and generates a lossless PNG image with the RGB mode. We don't generate lossy compression format. Our algorithm my not only be used for RGB JPEG or PNG images, but also can be applied on the other types of digital images.

The embedding process is based on the cellular automata with XOR local rule (Table 2). Cellular automata have been implemented to create the required secret key bit sequence. We only use eight numbers of the original image's features to generate this key. These values consist of sum of eigenvalues, sum of eigenvectors, mean of eigenvalues and mean of eigenvectors of the image.

TABLE 2. OUR PROPOSED CELLULAR AUTOMATA WITH XOR LOCAL RULE

Cell Number	Input Value	Rule
0	Sum of all *Eigen values* Numbers of *Red Matrix* of the Original Image.	
1	Mean of all *Eigenvalues* Numbers of *Red Matrix* of the Original Image.	
2	Sum of all *Eigenvectors* Numbers of *Red Matrix* of the Original Image.	$Cell[X]_{t+1} = Cell[X-1]_t \; XOR \; Cell[X+1]_t$
3	Mean of all *Eigenvectors* Numbers of *Red Matrix* of the Original Image.	
4	Sum of all *Eigenvalues* Numbers of *Green Matrix* of the Original Image.	
5	Mean of all *Eigenvalues* Numbers of *Green Matrix* of the Original Image.	
6	Sum of all *Eigenvectors* Numbers of *Green Matrix* of the Original Image.	
7	Mean of all *Eigenvectors* Numbers of *Green Matrix* of the Original Image.	

All of these values are easy to calculate and also exclusive for a particular matrix. Figure 2 shows the block diagram of the proposed method and Figure 3 illustrates the diagram of the proposed cellular automata to create a secret key base on these attributes of an image. We applied our proposed model on the input image of JPEG type, as shown in Figure 2, but our model is suitable and applicable for any format of RGB digital image.

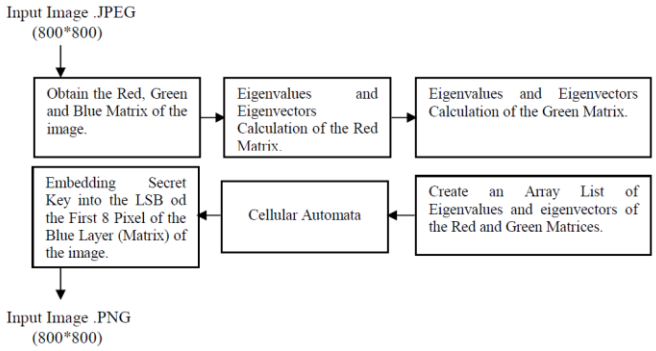

Figure 2. Block diagram of proposed model for digital images encryption in spatial domain.

The encryption algorithm in the frequency domain of the original image will be as follows:

Encryption Algorithm

***Input*:** .JPEG RGB image to apply our proposed data embedding on it for image encryption.

Output: .PNG RGB image file.

Step1: Open the original image and obtain the Red, Green and Blue matrices of the image.

Step2: Calculate the Eigenvalues and Eigenvectors of the Red Matrix.

Step3: Calculate the Eigenvalues and Eigenvectors of the Green Matrix.

***Step4*:** Perform the cellular automata rule according to the Table 2. This rule performs on the array list to create a Secret key.

Step5: Convert the Secret key to the binary representation.

***Step6*:** Select the first eight pixels in the Blue Layer (Blue Matrix) and embed the binary sequences of Secret key into the LSB of each pixel for encryption.

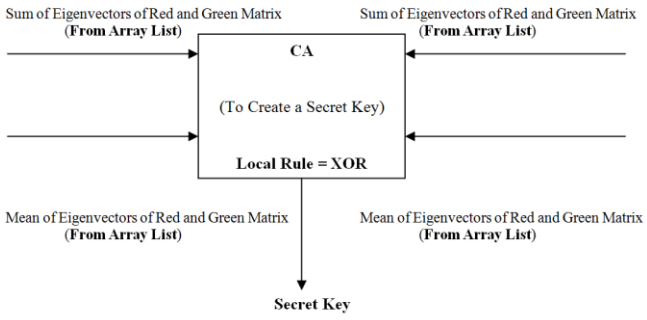

Figure 3. Block diagram of proposed cellular automata to create a secret key

The proposed algorithm has been implemented by C++ (C++ Builder XE2), which has the enough strength to work with digital images in any format.

http://sites.google.com/site/ijcsis/
ISSN 1947-5500

V. EXPERIMENTAL RESULTS

Four experimental results are given in this section to prove the performance and efficiency of the proposed model. These experimental results are as follows:

- Secret key sensitivity
- Diffusion
- Visual quality
- Time consumption

In order to evaluate the above aspects of our proposed method, we performed several tests on a sample dataset in our research laboratory. Our sample dataset contains 100 sets of RGB .JPEG images (size 800 × 800). All of our codes are implemented with C++ (C++ Builder XE2).

A. Secret key sensitivity

An ideal digital image encryption system should be sensitive with respect to the secret key. We mean a change of a single byte in the secret key should generate a completely different encrypted image and vice versa [10]. Table 3 shows the rate of secret key sensitivity.

TABLE 3. EVALUATION OF SECRET KEY SENSITIVITY AND ITS DEPENDENCY TO THE ORIGINAL IMAGE'S CHANGING

Image	Sum of Eigen Values (Red Layer)	Mean of Eigen Values (Red Layer)	
Sara	47	11	Original Image
	40	10	10 Pixels Changed
	38	9	20 Pixels Changed
Building	83	17	Original Image
	91	29	10 Pixels Changed
	87	21	20 Pixels Changed
Forest	69	19	Original Image
	82	13	10 Pixels Changed
	77	18	20 Pixels Changed

B. Diffusion

In the second experiment the diffusion of our secret key is considered. Diffusion means that the output bits should depend on the input bits in a very complex way. In a secret key with good diffusion, if one bit of the plaintext is changed, then the secret key should change completely [11]. Figure 4 shows the diffusion chart of our proposed model of generating the secret key.

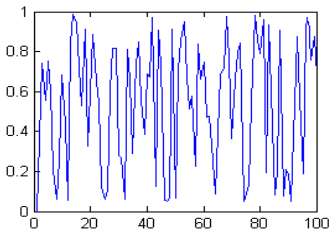

Figure 4. Diffusion Chart for Proposed Secret Key. Row indicates the number of images and column indicates the random number which is between 0 and 1.

C. Visual quality

We have generated a .PNG image as the output of our method with lossless compression. The experiment also used a .PNG image of size 800 × 800 as an input image. Figure 5 shows the original images and data embedded output PNG images which generated via our method to prove the visual quality of our method.

D. Time consumption

We select the first eight pixels of the blue layer to embed our secret key into the original image, as we mentioned in section 4. Table 4 presents the results of different pixels selection with different time consumption. It shows that minimum time consumption is obtained by embedding the secret key into the first eight pixels of the original image.

TABLE 4. EVALUATION OF SECRET KEY SENSITIVITY AND ITS DEPENDENCY TO THE ORIGINAL IMAGE'S CHANGING

	First 8 Pixels	Middle 8 Pixels	Last 8 Pixels
Time consumption for embedding the secret key	0.27	0.76	0.92

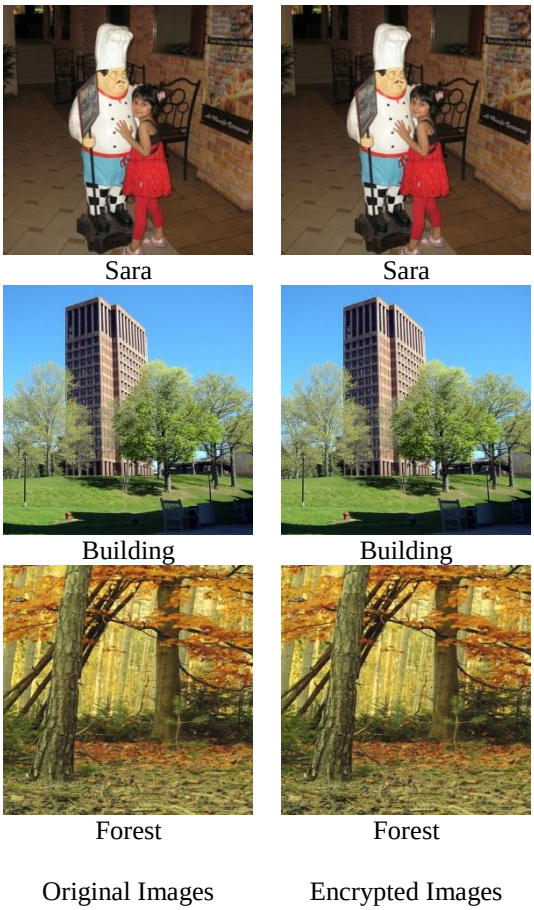

Figure 5. Encryption by the proposed algorithm and its visual quality

http://sites.google.com/site/ijcsis/
ISSN 1947-5500