Chapter 7
Active Image Forgery Detection Using Cellular Automata

Ahmad Pahlavan Tafti and Hamid Hassannia

Abstract. The adequate potential of digital images and the ease in their storage and distribution is such that they are more and more exploited to supply information in this digital epoch. As a consequence, they indicate a public source of evidence in our everyday life. Beside their benefits, the accessibility of them could bring a major detriment as they can be modified easily by a media processing application.

Detection of tampering with digital images is still an open work in the image processing domain. Over the past years there has been a swift expansion in the designing and developing of image forgery detection algorithms plus related software applications. All these algorithms are divided into two groups: active and passive. In the active approaches, we create and embed invaluable data as a cipher key into the original image to protect it against the forgery, while in the passive methods we only investigate some features of the image such as statistical anomalies, correlations and compressions to detect forgery.

This chapter presents an in-depth exploration of issues related to active digital image forgery detection algorithms which are derived from cellular automata. The aim of this chapter is to give a brief but comprehensive overview of the usage of cellular automata to develop active image forgery detection techniques. We conclude with experimental results in this topic and discuss future works in image forgery detection using cellular automata.

7.1 Introduction

We are living in an age where security of digital information such as digital images and videos are becoming more important than ever [5]. The expressive potential of

Ahmad Pahlavan Tafti
University of Wisconsin Milwaukee, USA
e-mail: pahlava2@uwm.edu

Hamid Hassannia
Department of Advanced Computing, Fan Pardaz Higher Education Institute, Iran
e-mail: hamid.h.h@ieee.org

visual media and the ease in their storage, and transmission are more and more exploited to convey information. Together with undoubted benefits, the accessibility of digital images brings a major drawback. With development of low cost, powerful image editing tools, the craft of tampering visual content is no more restricted to experts, so they can easily change image content and also it's meaning without leaving any traceable effect.

Image forgery detection methods in computer vision are quite able to authenticate the entire content of a digital image and protect them against tampering. A reliable images forgery detection system will be useful in many areas such as surveillance systems, medical imaging, criminal investigation, journalism, visa and immigration documents, insurance processing and forensic investigation.

The forgery detection techniques that are developed for digital images are mainly classified into two major classes, active and passive [2, 5] While in the active methods we would like to insert data or signature at the time of digitizing, the passive methods operate in the absence of any data or signature [2, 5]. In the active methods, we embed data into the original image to protect it against the forgery, where in the passive methods we don't have the original image and we should investigate some features such as statistical anomalies, correlations, compressions and measurements of objects in the existence image to detect forgery [4, 5].

Active approaches can be divided into two categories by the embedding in the position of spatial domain or frequency domain data [4]. Spatial domain techniques have already developed and are easier to implement but are limited in robustness [3]. Data embedding in the spatial domain consists of insertion and detection stages. The insertion algorithms are used to embed the data into the digital image and detection algorithms extract those data.

On validation and authentication aspects, the data which is embedded in a spatial domain should be unpredictable, invisible and also sensitive to any modification [3, 5].

In 2013 Anoop et al. [1] presented a full image encryption algorithm base on transform domain and stream cipher. In 2009 Krikor et al. [9] used DCT and stream cipher for digital image encryption. In 2003 Pommer et al. [13] provided an image encryption approach using selective encryption of wavelet packet. In 2003 Droogenbroeck et al. [19] developed Triple DES and IDEA based approach for the purpose of digital image encryption.

Using cellular automata as a discrete model is another way to generate such intricate information. This information would be embedded in a particular domain of an image for the purpose of image encryption and digital image forgery detection. In 2013 Xiaoyang et al. [20] used elementary cellular automata state rings to encrypt and decrypt QR code binary image. In 2012 Jin [8] developed an image encryption approach using the behavior of a number of Elementary Cellular Automata (ECA) with periodic boundary conditions. In 2011 Malakooti et al. [16] proposed a method to use the one dimensional cellular automata including statistical information of a digital image as an operational and practical way for image forgery detection. See also chapter 5.6 in this book which describes a passive method for copy-move forgery detection using cellular automata.

The goal of this chapter is to introduce such a framework, to propose cellular automata for implementing image forgery detection system and to give experimental results. In this chapter we proposed two active methods to detect digital image forgery in a reliable manner. The aim of this work is to develop a framework to active image forgery detection using cellular automata. The proposed methods take a digital image as input and compute some statistical information from its Lower Upper (LU) and Singular Value Decomposition (SVD). We use singular value decomposition and also lower upper decomposition plus one dimensional cellular automata to generate a cipher key. This key has the image features and completely related to digital image that every small change in the content of digital image will change the key value without any exception.

The rest of the chapter is arranged as follows. In Sections 7.2 and 7.3 we develop two different scenarios to image forgery detection based on a cellular automata. Section 7.4 introduces our sample dataset and describes the experimental results. Section 7.5 discusses limitations of the proposed models. Conclusion and areas for future development is considered in Section 7.6.

7.2 Scenario 1: Using Cellular Automata and LU Decomposition

LU decomposition is a kind of matrix decomposition which composes a matrix by the product of a lower and an upper triangular matrix [6]. Let A be a square matrix. An LU decomposition is a matrix decomposition of the form $A = LU$, where L and U are lower and top triangular matrices of the same dimensions. This means that L has just zeros overhead the diagonal and U has only zeros underneath the diagonal [6]. For a 3×3 matrix, this becomes:

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} = \begin{bmatrix} l_{11} & 0 & 0 \\ l_{21} & l_{22} & 0 \\ l_{31} & l_{32} & l_{33} \end{bmatrix} \begin{bmatrix} u_{11} & u_{12} & u_{13} \\ 0 & u_{22} & u_{23} \\ 0 & 0 & u_{33} \end{bmatrix}$$

Section 7.2.1 describes the usage of LU decomposition for the proposed active forgery detection algorithm. Let's a little describing one dimensional cellular automata we want to use in our proposed model. In this book, there are some chapters with details descriptions on cellular automata, so we just give a really brief overview which is necessary for the proposed model. Figure 7.1 shows a simple two state and one dimensional cellular automata with a line of cells. 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 [10, 17, 18].

We can define and set our own local rule for each cellular automata. You can see just two example of how we may define a local rule and how does it work. Let us to consider two following rules to estimate the value of cell X at time $t+1$ based on the value of cell X at the time t:

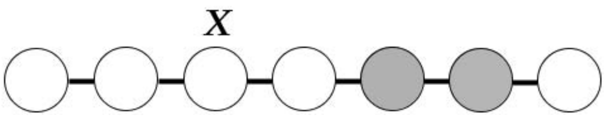

Fig. 7.1 One dimensional cellular automata with three neighborhoods for cell *X*

Rule 1: Cell[*X*] (*t*+1) = Cell[*X*-1] (*t*) (OR) Cell[*X*+1] (*t*)
Rule 2: Cell[*X*] (*t*+1) = Cell[*X*-1] (*t*) (AND) Cell[*X*+1] (*t*)

As long as we consider Rule 1, and the input sequence equals to 01100, then output sequence will be 11110. Table 7.1 shows the output of this cellular automata using an OR local rule.

By using Rule 2, while the input sequence equals to 01110, then output sequence will be 00100 (Table 7.2).

In this scenario, we propose a digital image forgery detection algorithm based on the cellular automata and LU decomposition. Experiments for this scenario will be described in detail in Section 7.4.

7.2.1 Proposed Model

The main idea of this proposed algorithm is to protect a digital image against forgery by creating and embedding an unpredictable cipher key into the spatial domain of an image. We embed the bit sequence of the cipher key into the LSB (Least Significant Bit) of the particular pixels in the original image because it takes less time on embedding. Section 7.4 shows some experimental validations obtained from embedding into different places. Using LSB decreases time consumption and may cause to reduce sensitivity to some attacks.

Our proposed algorithm performs on a grayscale images and generates a .png file including lossless PNG image with the grayscale model. We generate PNG image because generating any other formats like JPEG image would result into losing secret key embedded into pixel's LSB. The input type is not important in the proposed method and it may perform the same process on the different types of digital images such as RGB or CMYK, and different formats like .bmp, .gif, .pgm and etc. For

Table 7.1 An example of cellular automata

Cell Number	**0**	**1**	**2**	**3**	**4**
Input Sequence (time *t*)	0	1	1	0	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	0

Table 7.2 An example of cellular automata

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} (AND) Cell[X+1]^{t}				
Output Sequence (time $t+1$)	0	0	1	0	0

drawing a block diagram of our method (Figure 7.2), we just assume that the input type would be a .jpeg image.

Cellular automata have been implemented to create the required cipher key bit sequence. The XOR local rule used to generate the result in this chapter. We generate a cipher key by using specific cellular automata with an XOR local rule on six cells (see Table 7.3). In 2013 Sharma et al. [14] proposed a text security approach based on a XOR rule within 2D cellular automata and get well formed results. In 2011 Prasad Panda et al. [12] proposed a cellular automata encryption and decryption algorithm for block cipher based on XOR rules. Our experiments also show that a XOR logical operation often suffices to obtain a very sensitive cipher key. We have achieved a better rate for 'True alert' indicates true forgery detection, by using XOR rule rather than some other logical rules and also arithmetic rules (Table 7.6). Based on the experimental validations, our proposed XOR rule could improve PSNR (Table 7.7). Furthermore, it can be done extremely fast on contemporary CPUs that mostly provide a specific instruction to do a XOR operation [7].

We only use three number of statistical information of the LU decomposition matrices of the original image to generate the cipher key. This information consists of arithmetic mean, median, and the statistics range (Table 7.3). If anybody wants to modify a digital image, then the statistical information of these particular matrices will be changed, so the output of the proposed cellular automata will be damaged.

In this proposed method we consider single iteration cellular automata, implementing zero padding to obtain a valid value for boundary cells (Cells 0 and 5 in Table 7.3). We add zeros to both front and rear of our cellular automata so that the

Table 7.3 Proposed six cells for the cellular automata with XOR local rule: $Cell[X]^{t+1} = Cell[X-1]^{t}$ (XOR) $Cell[X+1]^{t}$

Cell Number	**Input Value (time t)**
0	Mean of the values in the L Matrix from LU decomposition of the original image
1	Mean of the values in the U Matrix from LU decomposition of the original image
2	Median of the values in the L Matrix from LU decomposition of the original image
3	Median of the values in the U Matrix from LU decomposition of the original image
4	Range of the values in the L Matrix from LU decomposition of the original image
5	Range of the values in the U Matrix from LU decomposition of the original image

rule could be applied to boundary cells. Since we are facing with the real input values (i.e. mean and range) for our proposed cellular automata, and we intend to apply XOR logical operation as a local rule, so we have to convert them to binary values. We foremost convert from real to integer values to avoid existing alteration complexities and limitations, applying recursive integer to binary alteration algorithm.

Finally, we embed the output of the proposed cellular automata into the LSB (Least Significant Bit) of the first eight pixels in the original image. LSB substitution is the process of modifying the least significant bit of the pixels of the input image. By doing this process, the value of a pixel is changed slightly, so the changes are not reflected physically in the output image. The proposed algorithm has been applied on grayscale images with pixel values between 0 and 255 in which the specific statistical information could be in the same range. Therefore, a byte should be enough to store output value as a cipher key. The experiments indicate that the LSB usage of the first eight pixels in this step is almost proper to obtain a well result on both time consuming and PSNR.

Here we briefly describe the statistical operations which are used in the proposed cellular automata.

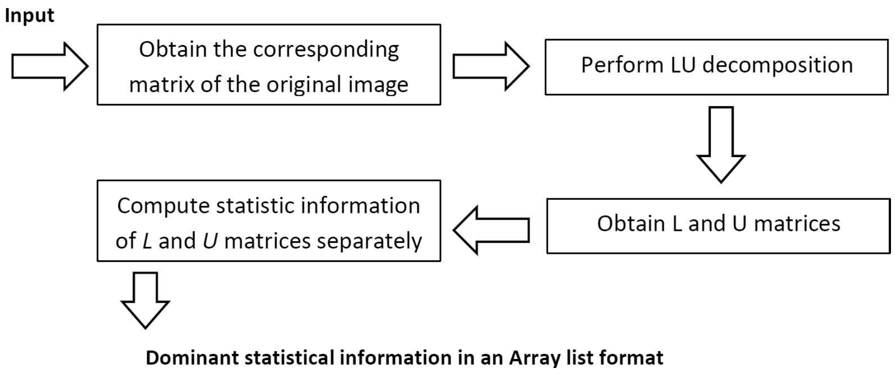
Input
Obtain the corresponding matrix of the original image
Perform LU decomposition
Obtain L and U matrices
Compute statistic information of *L* and *U* matrices separately
Dominant statistical information in an Array list format

Fig. 7.2 Block diagram of the proposed method

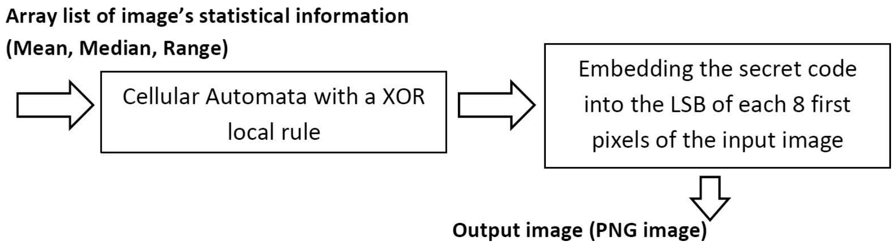
Array list of image's statistical information (Mean, Median, Range)
Cellular Automata with a XOR local rule
Embedding the secret code into the LSB of each 8 first pixels of the input image
Output image (PNG image)

Fig. 7.3 Block diagram of the proposed cellular automata for cipher key creation

Figure 7.2 shows the block diagram of the proposed method and Figure 7.3 illustrates the diagram of the proposed cellular automata to create a cipher key, based on statistical information of the L and U matrices.

The embedding algorithm in a spatial domain of the original image will be described in detail as follows:

Data Embedding Algorithm

Input: grayscale image to apply data embedding to it for forgery detection.

Output: .PNG grayscale image file.

Step1: Open the original image and get the corresponding matrix form of that. We know that a matrix is the certain underlying object of each digital image.

Step2: Perform the LU decomposition and obtain L and U matrices.

Step3: Calculate the statistical information for each L and U matrices separately, converting them to binary, and create the array list as the input values of the cellular automata.

Step 4: Zero padding to apply the rule to the boundaries.

Step5: Perform the cellular automata rule according to the Table 7.2. This rule performs on the array list to create a cipher key. (This step describes in detail in the following section: Forgery Detection Algorithm)

Step6: Convert the cipher keys to the binary representation.

Step7: Select the first eight pixels in the original image and embed the binary sequences of cipher key into the LSB of these eight pixels.

The forgery detection algorithm will be as follows:

Forgery Detection Algorithm

Input: .PNG image that contains the cipher key.

Output: Digital image forgery detection ALARM.

Step1: Open the .PNG input image and make corresponding digital image matrix.

Step2: initial integer variable CipherValue to zero.

Step3: initial integer variable PixelArrayValue to zero.

Step4: Perform the LU decomposition and obtain L and U matrices.

Step5: Calculate the statistical information for each L and U matrices separately and create the array list of these values.

Step6: Perform the cellular automata rule according to the Table 7.3. This rule performs on the array list to create a cipher key.

Step7: Select the first eight pixels of the image and extract the LSB binary value of pixels.

Step8: set CipherValue = value of the cipher key that generated in Step 6.

Step9: set PixelArrayValue = the extraction value in Step 7.

Step10: If PixelArrayValue = = CipherValue then print message ‘False Forgery Alarm’

Else Print message ‘True Forgery Alarm’;

This scenario is applied on grayscale images and it is definitely extendable to apply on color images.

7.3 Scenario 2: Using Cellular Automata and Singular Value Decomposition

Here, we define our second algorithm for digital image forgery detection. This part continues by providing a very brief description of Singular Value Decomposition (SVD), followed by our proposed model. Experiments for the second scenario will be also described in detail in Section 7.4.

SVD [6, 11] is very important in many areas of science. It is a way to very compactly represent what a matrix does to space. SVD can be seen as a generalization of eigenvalues and eigenvectors to a non-square matrix. It is very useful for solving linear algebraic problems like matrix inversion, linear least square estimation and fix-ranked approximation.

7.3.1 Proposed Model

The proposed method here is again based on the active approaches. Two set of dominant attributes that achieve from SVD (eigenvalues and eigenvectors) and one dimensional cellular automata could provide and generate the secret key. Here and in contrast to scenario 1, we prefer to work on RGB digital images to design such a scalable and flexible algorithm. We firstly achieve the eigenvalues and eigenvectors of the Red matrix (Red layer of the RGB image) of the input image and perform the same task for the Green matrix (Green layer of the RGB image). We calculate the eigenvalues and eigenvectors of the image, implementing one dimensional and single iteration cellular automata with a XOR local 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 first eight pixels of the Blue layer (Blue Matrix) in the original image. The reason of using XOR logical operation and first eight pixels for embedding purpose was illustrated in previous section. We need to use a real to binary conversion same as the scenario 1.

Table 7.4 shows the main idea of our proposed cellular automata. Only eight number of values of the original image have been used to generate this key. These values consist of sum of eigenvalues, sum of eigenvectors, mean of eigenvalues and mean of eigenvectors.

All of these values are easy to calculate and also exclusive for a particular matrix. Figure 7.4 shows the block diagram of the proposed method and Figure 7.5 illustrates the diagram of the proposed cellular automata to create a secret key, based on these attributes of an image. You see a .png format as an input image in this figure, but our proposed model is suitable and applicable for any format of RGB digital images.

Table 7.4 Proposed eight cells for the cellular automata with XOR local rule: $Cell[X]^{t+1} = Cell[X-1]^{t}$ (XOR) $Cell[X+1]^{t}$

Cell Number	Input Value (time t)
0	Sum of all singular values of the Original Image (Red Matrix)
1	Mean of all singular values of the Original Image (Red Matrix)
2	Sum of all right singular vectors of the Original Image (Red Matrix)
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

The embedding algorithm into LSB of eight pixels of the Blue layer of the input image will be as follows:

Data Embedding Algorithm

Input: RGB image to apply data embedding to it for forgery protection.

Output: .PNG RGB image file.

Step1: Open the original image and get the corresponding Red, Green and Blue matrices form of that. We know that a matrix is the certain underlying object of each digital image.

Step2: Calculate Eigenvalues and Eigenvectors of the Red Matrix and create the array list as the input values of the cellular automata.

Step3: Calculate Eigenvalues and Eigenvectors of the Green Matrix and create the array list as the input values of the cellular automata.

Step 4: Zero padding to apply the rule to the boundaries.

Step5: Perform the cellular automata rule according to the Table 7.4. This rule performs on the array list to create a Secret key. (This step describes in detail in the following section: Forgery Detection Algorithm)

Step6: Convert the Secret key to the binary representation.

Step7: Select the first eight pixels in Blue Layer (Blue Matrix) and embed the binary sequences of Secret key into the LSB of each pixel.

The forgery detection algorithm will be as follows:

Forgery Detection Algorithm

Input: .PNG image that contains a Secret key.

Output: Digital image forgery detection alarm.

Step1: Open the .PNG input image and make digital image matrix.

Step2: initial integer variable SecretValue to zero.

Step3: initial integer variable EigenVsArrayValue to zero.

Step4: Calculate Eigenvalues and Eigenvectors of the Red Matrix.

Step5: Calculate Eigenvalues and Eigenvectors of the Green Matrix.

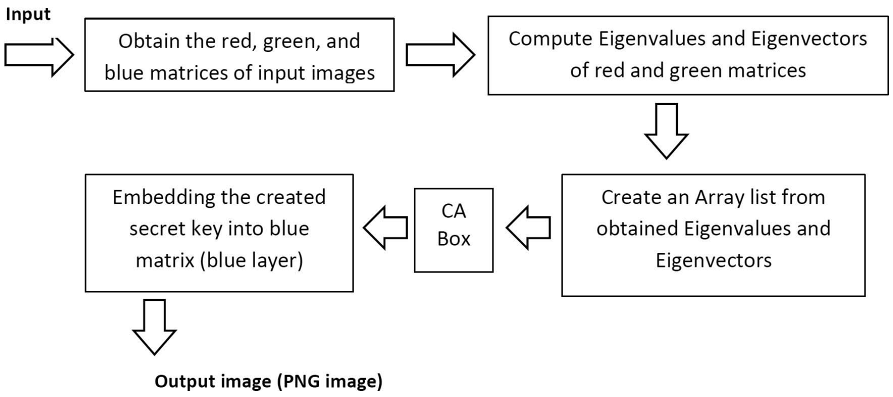
Input
Obtain the red, green, and blue matrices of input images
Compute Eigenvalues and Eigenvectors of red and green matrices
Create an Array list from obtained Eigenvalues and Eigenvectors
CA Box
Embedding the created secret key into blue matrix (blue layer)
Output image (PNG image)

Fig. 7.4 Block diagram of the proposed method

Step6: Perform the cellular automata rule according to the Table 7.2. This rule performs on the array list to create a Secret key.

Step7: Select the first eight pixels of Blue layer and extract the LSB binary value of each pixel.

Step8: set SecretValue = value of the Secret key that generated in Step 4 and 5 and 6.

Step9: set EigenVsArrayValue = the extraction value in Step 7.

Step10: If EigenVsArrayValue = = SecretValue then print message ‘False Forgery Alarm’

Else Print message ‘True Forgery Alarm’;

7.4 Dataset and Experimental Results

Table 7.5 shows our sample datasets and resources we have in our research laboratory in the Department of Advanced Computing, Fan Pardaz Higher Education Institute.

Table 7.5 Our dataset

Image File Format	.png and .jpeg
Image Type	Grayscale and RGB
Image Size	800*800 and 2560*1920
Number of Images	.png = 127 .jpeg = 149
Image Contents	Official digital documents, Medical images, Portrait

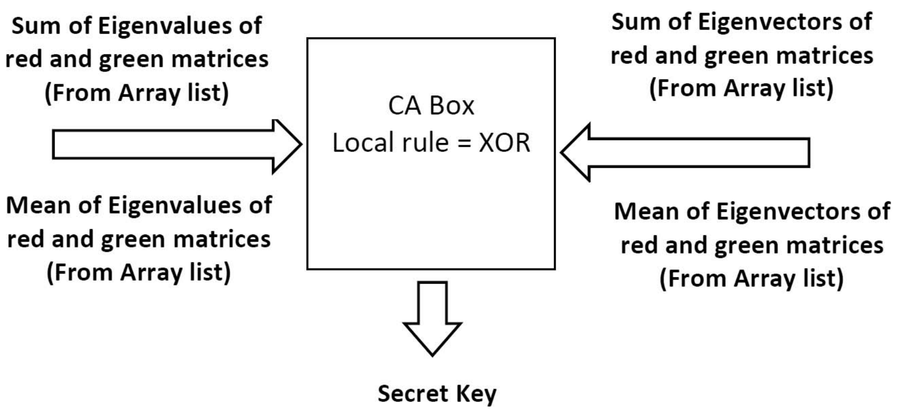
Sum of Eigenvalues of red and green matrices (From Array list)
Mean of Eigenvalues of red and green matrices (From Array list)
CA Box
Local rule = XOR
Sum of Eigenvectors of red and green matrices (From Array list)
Mean of Eigenvectors of red and green matrices (From Array list)
Secret Key

Fig. 7.5 Block diagram of the proposed cellular automata for cipher key creation

To prove the general performance of the proposed forgery detection methods, extensive experiments using real images were carried out. In particular, a configurable system has been built to implement both proposed algorithms. This system was built using JAVA SE with a simple and friendly user interface. All the experiments were carried out on a 3.00 GHz Intel Dual core 4MB cache with 4GB of RAM running 64-bit Windows 7 operating system. Seven experiments will be presented in this section to show the implementation and the results of the proposed methods. These are as follows:

- Performance and visual quality
- Time consumption
- True and False Alert
- PSNR
- Cipher key sensitivity
- Diffusion
- Comparison with non-CA works

In order to evaluate the above aspects of our proposed method, we perform several tests on the dataset (Table 7.5).

7.4.1 Performance and Visual Quality

Figure 7.6 shows the original images and data embedded output .png images which generated via scenario 1 to prove the performance and visual quality of the proposed method. By considering same images, input and output images using scenario 2 are shown in Figure 7.7.

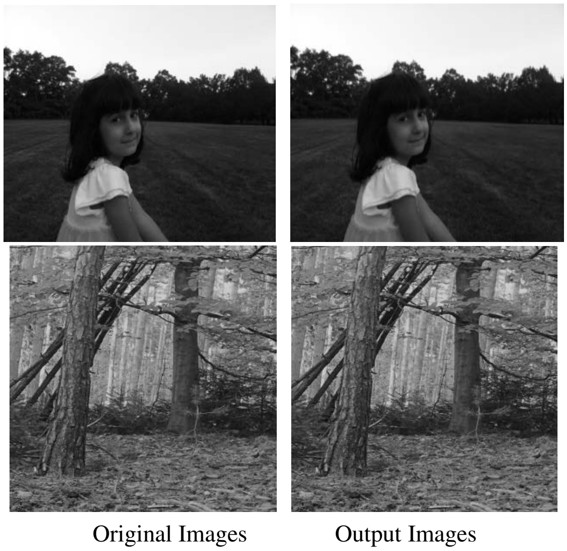

Fig. 7.6 Scenario 1: input and output images

7.4.2 Time Consumption

We select the first eight pixels of the original image to embed the proposed cipher, as we mentioned in Sections 7.2 and 7.3. Figure 7.8 presents the results of different pixels selection with different time consumption in which 50 images for each scenario separately. Figure 7.9 shows the same results with using 100 images. In this experiment we only used 50 and 100 images from the dataset and this number of images is only a random selective set of digital images we had. These figures show that minimum time consumption is obtained by embedding the cipher key into the first eight pixels of the original image.

7.4.3 True and False Alert

In Table 7.6, you can see the percentage of true and false detection of digital image forgery which are performed by the proposed method. We compare these two indices with different local rule for proposed cellular automata in Sections 7.2 and 7.3, plus different position to embed data into the original image. We use exactly same images for both scenarios. In this experiment, three various logical operations (XOR, AND, and OR) and two different arithmetic operations (Addition and Multiply) are implemented as our cellular automata's local rule.

As shown in this table, and by considering previous results (Time consumption), scenario 2 using XOR local rule with data embedding in only first eight pixels of the original images are better in average.

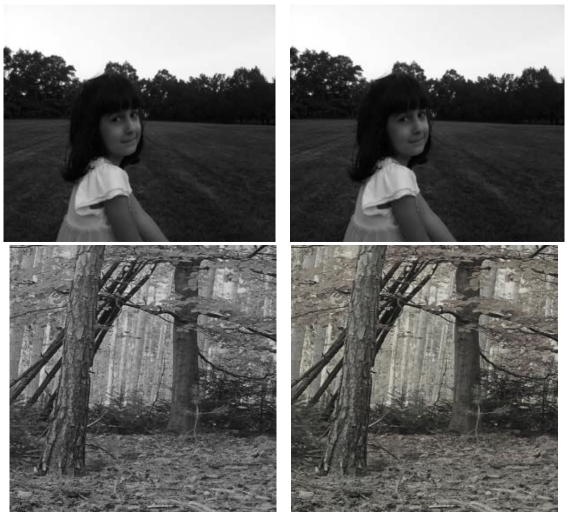

Fig. 7.7 Scenario 2: input and output images

7.4.4 PSNR

Measuring the PSNR (peak signal-to-noise ratio) of our proposed method is the next experiment. It indicates the maximum possible power of a signal and the power of corrupting noise that affects the output. We mentioned that all pixels in both of the input and output images in our proposed method are based on 8 bits. The result of PSNR for our proposed algorithm is shown in Table 7.7. Typical values for the PSNR in lossy image and video compression are between 30 and 50 dB, where higher is better [3], therefore, our proposed algorithm has a good PSNR.

This experiment indicates that scenario 1 produces a very little better PSNR than scenario 2. This kind of experiment in the dataset indicates that using logical operation instead of arithmetic operation was not generating quite different result regarding the PSNR.

7.4.5 Secret Key Sensitivity

An ideal digital image encryption system should be sensitive with respect to the secret key. A little change of a single byte in the secret key should generate a completely different encrypted image and vice versa. Table 7.8 shows the rate of secret key sensitivity. Previous experiments indicate that scenario 2 is better than scenario 1 in average, so in this experiment we just focus on scenario 2. 'Sara' is an image which is shown at the first row of Figure 7.7 and 'Forest' is the second one. Since one of the our goal is to develop sensitive cipher key for image forger detection

Table 7.6 True and false detection of the proposed approaches. Three different logical operations and two arithmetic operations are used to evaluate a local rule of the proposed cellular automata. We use a hundred images for both scenarios.

Approach	**True Alert %**	**False Alert %**
Scenario 1: XOR operation + embedding into first eight pixels	94.61	5.39
Scenario 1: XOR operation + embedding into first and middle eight pixels	94.61	5.39
Scenario 1: XOR operation + embedding into first, middle and end eight pixels	94.79	5.21
Scenario 1: AND operation + embedding into first eight pixels	88.67	11.33
Scenario 1: AND operation + embedding into first and middle eight pixels	88.68	11.32
Scenario 1: AND operation + embedding into first, middle and end eight pixels	88.71	11.29
Scenario 1: OR operation + embedding into first eight pixels	88.29	11.71
Scenario 1: OR operation + embedding into first and middle eight pixels	88.29	11.71
Scenario 1: OR operation + embedding into first, middle and end eight pixels	88.32	11.68
Scenario 1: Addition operation + embedding into first eight pixels	81.44	18.56
Scenario 1: Addition operation + embedding into first and middle eight pixels	81.47	18.53
Scenario 1: Addition operation + embedding into first, middle and end eight pixels	81.50	18.50
Scenario 1: Multiply operation + embedding into first eight pixels	91.14	8.86
Scenario 1: Multiply operation + embedding into first and middle eight pixels	91.15	8.85
Scenario 1: Multiply operation + embedding into first, middle and end eight pixels	91.19	8.81
Scenario 2: XOR operation + embedding into first eight pixels	96.23	3.77
Scenario 2: XOR operation + embedding into first and middle eight pixels	96.25	3.75
Scenario 2: XOR operation + embedding into first, middle and end eight pixels	96.29	3.71
Scenario 2: AND operation + embedding into first eight pixels	86.09	13.91
Scenario 2: AND operation + embedding into first and middle eight pixels	86.09	13.91
Scenario 2: AND operation + embedding into first, middle and end eight pixels	86.17	13.83
Scenario 2: OR operation + embedding into first eight pixels	86.01	13.99
Scenario 2: OR operation + embedding into first and middle eight pixels	86.03	13.97
Scenario 2: OR operation + embedding into first, middle and end eight pixels	86.08	13.92
Scenario 2: Addition operation + embedding into first eight pixels	84.73	15.27
Scenario 2: Addition operation + embedding into first and middle eight pixels	84.73	15.27
Scenario 2: Addition operation + embedding into first, middle and end eight pixels	84.77	15.23
Scenario 2: Multiply operation + embedding into first eight pixels	92.32	7.68
Scenario 2: Multiply operation + embedding into first and middle eight pixels	92.32	7.68
Scenario 2: Multiply operation + embedding into first, middle and end eight pixels	92.37	7.63

purposes, we can examine how pixel's value changes results appear in related values of the cipher key. This table shows that even small changes in pixel values of the original image can make a different value for the cipher key.

In this scenario, we propose a digital image forgery detection algorithm based on the cellular automata and LU decomposition. Experiments for this scenario will be described in detail in Section 7.4.

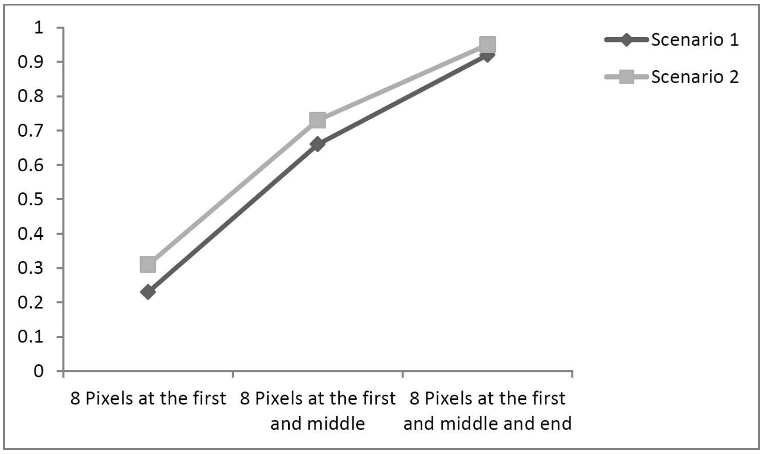

Fig. 7.8 Time consumption for embedding the cipher key into the original image. Here, we used 50 digital images for each scenario. Vertical axis shows the time consumption in seconds.

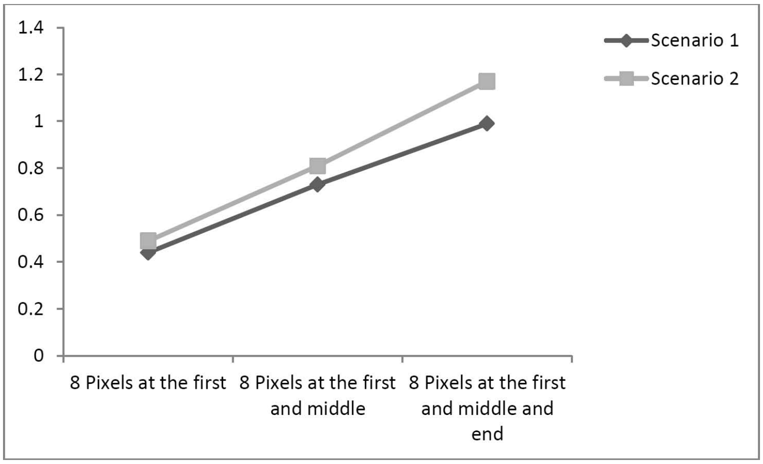

Fig. 7.9 Time consumption for embedding the cipher key into the original image. Here, we used 100 digital images for each scenario. Vertical axis shows the time consumption in seconds.

7.4.6 Diffusion

In this 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 [15]. More generally, one may require that flipping a fixed set of bits should change each output bit with probability one half.

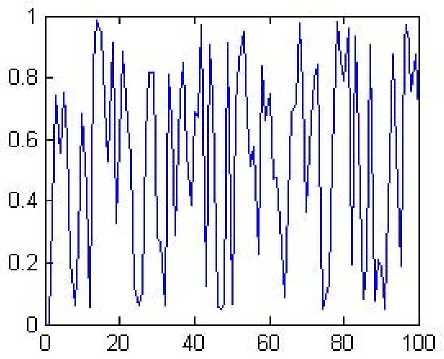

Fig. 7.10 Diffusion Chart for Proposed Secret Key (Average in both scenarios). Row indicates the number of images and column indicates the random number which is between 0 and 1.

Figure 7.10 shows the diffusion chart of our proposed model of generating the secret key. We only examine XOR local rule for this diffusion evaluation. This figure exposes high relative distribution index from 0 to 1, indicating well-founded diffusion for the proposed secret key.

7.4.7 Comparison with Non-CA Works

In Table 7.9, we compared our proposed method with other non-CA algorithms mentioned in literature review (Section 7.1).

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. The proposed algorithm in this chapter generates visible images (Figure 7.6 and Figure 7.7) and it is not high-level encryption method in which we will face with really disordered image.

Our proposed CA approach has used the internal information of a digital image instead of the some logo and external information. Using external logo or other data may cause to exceed the size of the output images.

7.5 Limitations

The proposed cellular automata's are only one dimensional and it could be considered as a limitation for our system. The other limitation is using active approach which is needed the original image for forgery detection. Similarly, we do not explicitly model noise as an external effect. The main limitation of the proposed method is sensitivity to post processing.

Table 7.7 PSNR evaluation of the proposed approaches. Three different logical operations and two arithmetic operations are used in this experiment. We used a hundred images for both scenarios, putting the average in this table.

Approach	**PSNR**
Scenario 1: (XOR operation + embedding into the first and middle eight pixels	39.11
Scenario 1: XOR operation + embedding into first, middle and end eight pixels	39.04
Scenario 1: AND operation + embedding into first eight pixels	39.25
Scenario 1: AND operation + embedding into first and middle eight pixels	39.21
Scenario 1: AND operation + embedding into first, middle and end eight pixels	39.01
Scenario 1: OR operation + embedding into first eight pixels	39.27
Scenario 1: OR operation + embedding into first and middle eight pixels	39.22
Scenario 1: OR operation + embedding into first, middle and end eight pixels	39.16
Scenario 1: Addition operation + embedding into first eight pixels	37.76
Scenario 1: Addition operation + embedding into first and middle eight pixels	37.58
Scenario 1: Addition operation + embedding into first, middle and end eight pixels	37.51
Scenario 1: Multiply operation + embedding into first eight pixels	37.69
Scenario 1: Multiply operation + embedding into first and middle eight pixels	37.57
Scenario 1: Multiply operation + embedding into first, middle and end eight pixels	37.42
Scenario 2: XOR operation + embedding into first eight pixels	38.13
Scenario 2: XOR operation + embedding into first and middle eight pixels	38.07
Scenario 2: XOR operation + embedding into first, middle and end eight pixels	37.92
Scenario 2: AND operation + embedding into first eight pixels	38.19
Scenario 2: AND operation + embedding into first and middle eight pixels	38.11
Scenario 2: AND operation + embedding into first, middle and end eight pixels	38.05
Scenario 2: OR operation + embedding into first eight pixels	38.28
Scenario 2: OR operation + embedding into first and middle eight pixels	38.16
Scenario 2: OR operation + embedding into first, middle and end eight pixels	38.08
Scenario 2: Addition operation + embedding into first eight pixels	36.14
Scenario 2: Addition operation + embedding into first and middle eight pixels	36.10
Scenario 2: Addition operation + embedding into first, middle and end eight pixels	36.01
Scenario 2: Multiply operation + embedding into first eight pixels	36.36
Scenario 2: Multiply operation + embedding into first and middle eight pixels	36.27
Scenario 2: Multiply operation + embedding into the first, middle and end eight pixels)	36.12

Table 7.8 Evaluation of secret key sensitivity and its dependency to the original image's changing

Image	Manipulation	Sum of Eigenvalues (Red Layer)	Mean of Eigenvalues (Red Layer)
Sara	Original Image	51	16
	10 Pixels Changed	43	11
	20 Pixels Changed	38	9
Forest	Original Image	67	21
	10 Pixels Changed	81	15
	20 Pixels Changed	73	14

Table 7.9 Comparison with non-CA algorithms

Developers	**Algorithm**	**Data or items used**	**Encryption level**
Van Droogenbroeck et al. [19]	Triple DES and IDEA	External Logo	high-level
Pommer et al. [13]	Selective Encryption of Wavelet-Packet	External Logo	high-level
Krikor et al. [9]	DCT and Stream Cipher	Pseudo-Random bit sequence. (External Key)	high-level
Anoop et al. [1]	Transform Domains and Stream Ciphers	Stream RC4 key values	high-level
models proposed in this chapter	1D Cellular Automata	Internal properties of the input image	low-level

7.6 Conclusion and Future Work

A cellular automata approach presented here for active image forgery detection. In this chapter we have presented one scenario based on LU decomposition and other scenario based on the SV decomposition with combination of one dimensional cellular automata. The cellular automata rule generates a cipher key which can be used to embed into the image. Both procedures need the original images to notice forgery detection.

Seven different experimental validations have also been done. These experimental results obtained from the methods, specially the diffusion and true and false alert clearly shown the performance and reliability of the models. In future work, we will aim on two dimensional cellular automata forms into the suggested framework.

References

1. Anoop, S., Alakkaran, A.: A full image encryption scheme based on transform domains and stream ciphers. International Journal of Advanced Information Science and Technology 17(17), 5–10 (2013)
2. Birajdar, G.K., Mankar, V.H.: Digital image forgery detection using passive techniques: A survey. Digital Investigation 10(3), 226–245 (2013)
3. Bovik, A.C.: The essential guide to image processing. Academic Press (2009)
4. Cox, I., Miller, M., Bloom, J., Fridrich, J., Kalker, T.: Digital Watermarking and Steganography, 2nd edn. Morgan Kaufmann Publishers Inc., San Francisco (2007)
5. Farid, H.: Image forgery detection. IEEE Signal Processing Magazine 26(2), 16–25 (2009)
6. Golub, G.H., van Van Loan, C.F.: Matrix computations (Johns Hopkins studies in mathematical sciences), 3rd edn. The Johns Hopkins University Press (1996)
7. Intel: Sse4 programming reference. D91561 (2007)
8. Jin, J.: An image encryption based on elementary cellular automata. Optics and Lasers in Engineering (2012)
9. Krikor, L., Shaaban, Z.: Image encryption using DCT and stream cipher. European Journal of Scientific Research 32(1), 47–57 (2009)
10. Lafe, O.: Data compression and encryption using cellular automata transforms. Engineering Applications of Artificial Intelligence 10(6), 581–591 (1997)
11. Mao, K.: Identifying critical variables of principal components for unsupervised feature selection. IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics 35(2), 339–344 (2005)
12. Panda, S.P., Sahu, M., Rout, U.P., Nanda, S.K.: Encryption and decryption algorithm using two dimensional cellular automata rules in cryptography. International Journal of Communication Network & Security 1, 18–23 (2011)
13. Pommer, A., Uhl, A.: Selective encryption of wavelet-packet encoded image data: efficiency and security. Multimedia Systems 9(3), 279–287 (2003)
14. Sharma, P., Lal, N., Diwakar, M.: Text security using 2d cellular automata rules. In: Proceedings of the Conference on Advances in Communication and Control Systems 2013. Atlantis Press (2013)
15. Stallings, W.: Cryptography and Network Security, 3rd edn. Practice Hall (2003)
16. Tafti, A.P., Malakooti, M., Ashourian, M., Janosepah, S.: Digital image forgery detection through data embedding in spatial domain and cellular automata. In: Int. Conf. on Digital Content, Multimedia Technology and its Applications (IDCTA), pp. 11–15 (2011)
17. Toffoli, T., Margolus, N.: Cellular Automata Machines: A new environment for modelling. MIT press (1987)
18. Urias, J.: Cryptography primitives based on a cellular automaton. In: Coding Theory, Cryptography and Related Areas, pp. 244–248 (2000)
19. Van Droogenbroeck, M., Benedett, R.: Techniques for a selective encryption of uncompressed and compressed images. In: Proceedings of the ACIVS Advanced Concepts for Intelligent Vision Systems (2002)
20. Xiaoyang, Y., Yang, S., Yang, Y., Shuchun, Y., Hao, C., Yanxia, G.: An encryption method for QR code image based on ECA. International Journal of Security & Its Applications 7(5) (2013)