International Journal of Engineering Research and General Science Volume 4, Issue 1, January-February, 2016
ISSN 2091-2730

Data Hiding In Encrypted Images

Chennatt Padmanabhan Suparna

Kannur University, suparnapadmanabhan@gmail.com and 09746146024

Abstract—Recently more attention is paid to reversible data hiding in encrypted images, since it maintains the excellent property that the original cover can be losslessly recovered after the embedded data is extracted while protecting the image content confidentiality. All previous methods embed data by reversibly vacating room from the encrypted images, which may be subject to some errors on data extraction and image restoration. Here propose a novel method by reserving room before encryption with a traditional algorithm and thus it is easy for the data hider to reversibly embed the data in the encrypted image. The proposed method can achieve real reversibility that is data extraction and image recovery are free of any error.

Keywords— reversible data hiding (RDH), image encryption, encryption key, data hiding ke, image decryption, privacy protection , rationale rhombus, LSB replacement, vacating room after encryption (VRAE), reserving room before encryption (RRBE).

INTRODUCTION

The amount of digital images has been increased rapidly, therefore the protection of multimedia data is very important for many applications, such as confidential transmission, video surveillance, military and medical field applications. To decrease the transmission time, the data compression is necessary. The protection of multimedia data can be done with the compression, encryption and data hiding. Two main groups of technologies have been developed for this purpose. The first one is based on content protection through encryption. The second group bases the protection on data hiding, aimed at secretly embedding a message into the data. These two technologies can be used complementary and mutually commutative. Major objective of the work is reserving room before encryption with a traditional RDH algorithm, and thus it is easy for the data hider to reversibly embed data in the encrypted image. Thus it achieves real reversibility, that is data extraction and image recovery are free of any error. Thus the data hider can benefit from the extra space emptied out in previous stage to make data hiding process effortless. Previous work proposed to embed data in an encrypted image by using an irreversible approach of data hiding. But the possibility of noise contained in the decrypted image. Now the challenge was to find an encryption method robust to noise. This problem can be resolved by the proposed method, reversible data hiding in encrypted images by reserving room before encryption. An adaptive median filter is also applied at the final stage of decryption to remove the impulse noise contents in the image.

SYSTEM DESIGN AND IMPLEMENTATION

Data Hiding is the technique of hiding the data into cover media. The data hiding process involve two sets of data, first one is a set of embedded data and another is a set of the cover media data. As when data is embedded into the image then the quality of image get disturbed. So it is expected that after the data extraction the image quality should be maintained just like the original image. But that image contains some distortions. RDH in images is a technique, due to which the original cover can be loss less recovered after the embedded message is extracted. The previous methods can be summarized as the framework, vacating room after encryption (VRAE). Vacating room from the encrypted images losslessly is relatively difficult and also sometimes inefficient. Thus, reverse the order of encryption and vacating room, that is reserving room prior to image encryption at content owner side, the RDH tasks in encrypted images would be more natural and are much easier which leads to the framework, reserving room before encryption (RRBE). As shown in Figure 1, the content owner first reserves enough space on original image and then convert the image into its encrypted version with the encryption key. The data embedding process in encrypted images is inherently reversible for the data hider only needs to accommodate data into the spare space previous emptied out. The data extraction and image recovery are performed by the receiver. The double encrypted image containing data is first decrypted by using the data hiding key and extracts the data stored in the image. Then it is decrypted by using the encryption key and recovers the original image. At the decryption side there is a possibility of noise that is any undesirable signal. Noise gets introduced into the data via any electrical system used for storage, transmission, or any other processing. In order to remove the noise content in the decrypted image an adaptive median filter is used at the final stage. The adaptive median filter performs spatial processing to determine which pixels in an image have been affected by impulse noise. It helps

to remove impulse noise and smooth other noise, which in turn reduce distortion, like excessive thinning or thickening of object boundaries.

Figure 1. Block diagram of data hiding in encrypted image.

Elaborate a practical method based on this Framework, which primarily consists of four stages: generation of encrypted image, data hiding in encrypted image, data extraction and image recovery.

a) Generation of encrypted image

To construct the encrypted image, the first stage can be divided into three steps: image partition, self reversible embedding followed by image encryption. At the beginning, image partition step divides original image into two parts A and B; then, the LSBs of A are reversibly embedded into B using rationale rhombus algorithm so that LSBs of A can be used for accommodating messages.

1.Image Partition

The operator here for reserving room before encryption is a standard reversible data hiding technique, so the goal of image partition is to construct a smoother area on which rationale rhombus algorithms can be used to achieve better performance. To do that, without loss of generality assume the original image is an 8 bits gray scale image with its size M×N and pixels Ci,j ϵ [0,255],1 ≤ i ≤ M ≤ j ≤ N. First, the content owner extracts from the original image, along the rows, several overlapping blocks whose number is determined by the size of embedded messages. In detail, every block consists of m rows, where, m = [l/N] and the number of blocks can be computed through n= M-m+1. An important point here is that each block is overlapped by pervious or sub sequential blocks along the rows. For each block define a function to measure its first order smoothness. $f = \sum_{u=2}^{m}\sum_{v=2}^{N-1}\left| C_{u,v} - \frac{C_{u-1,v} + C_{u+1,v} + C_{u,v-1} + C_{u,v+1}}{4}\right|$ Higher relates to blocks which contain relatively more complex textures. The content owner, selects the particular block with the highest to be A, and puts it to the front of the image concatenated by the rest part with fewer textured areas, as shown in Figure 2.

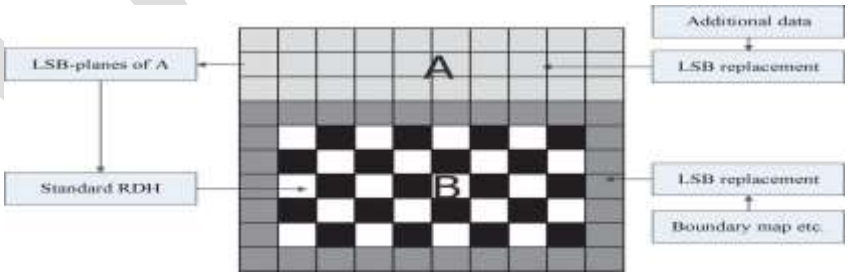

Figure 2. Illustration of image partitioning and embedding process.

2. Self-Reversible Embedding

The goal of self reversible embedding is to embed the LSB planes of A into B by employing traditional RDH algorithms. For illustration, simplify the method in to demonstrate the process of self embedding. This step does not rely on any specific RDH

algorithm. Pixels in the rest of image B are first categorized into two sets: white pixels with its indices i and j satisfying (i + j) mod2= 0 and black pixels whose indices meet (i + j) mod2 = 1, as shown in Figure. 2. Then, each white pixel $B_{i,j}$ is estimated by the interpolation value obtained with the four black pixels surrounding it is $\mathbf{B'_{i,j} = \omega_1 B'_{i-1,j} + \omega_2 B'_{i+1,j} + \omega_3 B'_{i,j-1} + \omega_4 B'_{i,j+1}}$ Where the weight ω_i $1 \leq i \leq 4$, the estimating error is calculated via $e_{i,j} = B_{i,j} - B'_{i,j.}$ and then some data can be embedded into the estimating error sequence. Further calculate the estimating errors of black pixels with the help of surrounding white pixels that may have been modified. Then another estimating error sequence is generated which can accommodate messages and can also implement multilayer embedding scheme by considering the modified B as original one when needed. In summary, to exploit all pixels of B, two estimating error sequences are constructed for embedding messages in every single layer embedding process.

3. Image Encryption

After rearranged self embedded image and reserving rooms, denoted by X , is generated. Then encrypt X to construct the encrypted image, denoted by E. Encryption key is an 8 bit key. In this the encryption is done by XORing the image with the key. With a stream cipher, the encryption version of X is easily obtained. For example, a gray value ranging from 0 to 255 can be represented by 8 bits, $X_{i,j}(0)$, $X_{i,j}(1)$, $X_{i,j}(2)$..... $X_{i,j}(7)$ Such that $X_{i,j}(k) = \frac{X_{i,j}}{2^k} \bmod 2$, $k = 0,1, \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots.7.$

The encrypted bits can be calculated through exclusive or operation, $E_{i,j(k)} = X_{i,j} \oplus r_{i,j}$.

Finally, embed 10 bits information into LSBs of first 10 pixels in encrypted version of to tell data hider the number of rows and the number of bit planes that can embed information into encrypted image. Note that after image encryption, the data hider or a third party cannot access the content of original image without the encryption key, thus privacy of the content owner being protected. After all this steps the content owner sends the locations where the data can be embedded along with the encrypted image.

b) Data hiding in encrypted image

Once the data hider acquires the encrypted image can embed some data into it, although he does not get access to the original image. The embedding process starts with locating pixels in which the data can embed in the encrypted version of image. Since the data hider has the locations where the data can be embedded it is effortless for the data hider to read bits information in LSBs of encrypted pixels. After knowing how many bit planes and rows of pixels he can modify, the data hider simply adopts LSB replacement to substitute the available bit planes with additional data. Finally, the data hider encrypts according to the data hiding key to formulate encrypted image containing data.

c) Data extraction and image recovery

In this case the two keys that is the encryption key and the data hiding key, both are used to extract the data and then the image. Since the image is encrypted two times, the image can be restored only after the data extraction. Hence first using the data hiding key the image is decoded and the data stored in the LSBs of the pixels in the image is extracted. Then the decoded image once again decoded with the encryption key. Then by applying the rationale rhombus algorithm for data extraction the LSB values are restored and placed it in its own position using LSB replacement algorithm. Finally the extracted image is passed through the adaptive median filter that helps to reduce the impulse noise and also smooth all other types of noise. The adaptive median filter preserves detail and smooth non impulsive noise. Hence it provide noiseless decrypted output image.

ALGORITHM

i. Rationale Rhombus Algorithm

Rationale rhombus method used to store the LSB values of the pixel of the portion of the image. Rationale rhombus method is implemented here to effectively hide and make is available in the decoding time. In an image the adjacent pixels have the pixel value in less difference. So while considering a rhombus in the pixels the pixel cantered by the four pixels has the average value of the four pixels. This technique is used to develop the rationale rhombus algorithm. These pixels are classified as Cross and Dot. Dots are the four pixels used to find the average value and its value is not changed in this method. Cross is the pixel where the data is hiding and is modified with the following algorithm. Henceforth, this scheme will be called the Cross embedding scheme.

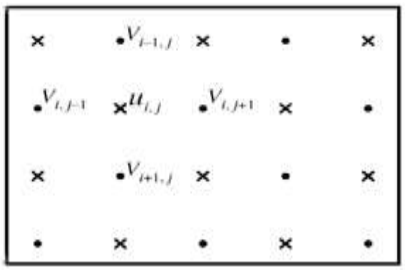

Figure 3. Prediction pattern

In order to predict pixel value of position ui, j in Fig.3, four neighbouring pixels "Dots" are used. Vi, j−1, vi+1, j, vi, j+1, and vi−1, j. Ui,j is used as the Cross to store the data. The five pixels including ui, j comprises a cell which is used to hide one bit of data.

The encoder of the Cross embedding scheme for a single cell is as follows. Centre pixel ui, j of the cell can be predicted from the four neighbouring pixels vi, j−1, vi+1, j, vi, j+1, and vi−1, j.

The predicted value u'i, j is computed as $u'i,j = [vi,j-1 + vi+1,j + vi,j+1 + vi-1,j/4]$.

Based on the predicted value u'i, j and original value ui, j, the prediction error "E" is computed as E = ui, j – u'i, j and M = 2E.

Prediction error can be expanded to hide information as H = M + bit Where H is the prediction error after expansion called modified prediction error. The bit is the hiding data. After data hiding, the original pixel value ui, j is changed to Ui, j as Ui, j = H + u'i, j.

The decoding procedure for the Cross embedding scheme for a single cell is an inverse of the encoding scheme. During data hiding, pixels from the Dot set are not modified, so the predicted values u'i, j is also not changed. Using the predicted value u'i, j and the modified pixel value Ui, j, and the decoder can exactly recover the embedded bit and original pixel value.

The modified prediction error is computed as H = Ui, j – u'i, j.

The embedded bit value is computed as bit = H mod 2.

The original prediction error is computed as M = H – bit and E = M / 2.

The original pixel's value is computed as ui, j = u'i, j + E Note that the two sets (the Cross set and Dot set) are independent of each other. Independence means changes in one set do not affect the other set, and vice versa. Pixels from the Dot set are used for computing predicted values u'i, j, whereas pixels from the Cross set ui, j are used for embedding data. The order of hiding data in cells is not important and can be changed. Sorting reorders cells according to the magnitudes of local variance and enables hiding data in cells with small prediction errors. Thus, sorting can significantly improve the data embedding scheme.

ii. LSB Replacement Algorithm

In this algorithm embed the each bit of the data in the least significant bits places of the original image. The embedding of the data is performed choosing a subset of image pixels and substituting the least significant bit of each of the chosen pixels with embedding bits. The extraction of the data is performed by extracting the least significant bit of each of the selected image pixels. If extracted bits match the inserted bits, then the stored is detected. The extracted bits do not have to exactly match with the inserted bits. A correlation measure of both bit vectors can be calculated. If the correlation of extracted bits and inserted bits is above a certain threshold, then the extraction algorithm can decide that the data is detected.

Step 1. Load the original image.

Step 2. Load the embedding data.

Step 3. Determine the value of the embedding factor

Step 4. Call the embedding function to embed the bits in the least significant bits of the original image.

Step 5. Use the extraction function to extract the watermark.

iii. Adaptive Median Filtering Algorithm

The Adaptive Median Filter performs spatial processing to determine which pixels in an image have been affected by impulse noise. The Adaptive Median Filter classifies pixels as noise by comparing each pixel in the image to its surrounding neighbor pixels. The size of neighborhood is adjustable, as well as the threshold for the comparison. A pixel that is different from a majority of its neighbors, as well as being not structurally aligned with those pixels to which it is similar, is labeled as impulse noise. These noise pixels are then replaced by the median pixel value of the pixels in the neighborhood that have passed the noise labeling test. Adaptive median filter changes size of Sxy (the size of the neighborhood) during operation.

● Notation

Zmin = minimum gray level value in Sxy
Zmax = maximum gray level value in Sxy
Zmed = median of gray levels in Sxy
Zxy = gray level at coordinates (x, y)
Smax = maximum allowed size of Sxy

● Algorithm

Level A : A1 = Zmed – Zmin
A2 = Zmed – Zmax
if A1 > 0 AND A2 < 0, go to level B
else increase the window size
if window size < Smax, repeat level A
else output Zxy
Level B: B1 = Zxy - Zmin

B2 = Zxy – Zmax
if B1 > 0 AND B2 < 0, output Zxy
else output Zmed

● Explanation

Level A: IF Zmin < Zmed < Zmax, then
• Zmed is not an impulse
(1) go to level B to test if Zxy is an impulse .
ELSE
• Zmed is an impulse
(1) the size of the window is increased and
(2) level A is repeated until .
(a) Zmed is not an impulse and go to level B or
(b) Smax reached: output is Zxy

Level B: IF Zmin < Zxy < Zmax, then
• Zxy is not an impulse
(1) output is Zxy (distortion reduced)
ELSE
• either Zxy = Zmin or Zxy = Zmax
(2) output is Zmed (standard median filter)
• Zmed is not an impulse (from level A)

HARDWARE AND SOFTWARE REQUIREMENTS

1. Software requirements

a) MATLAB

Matlab is a high performance language for technical computing. It integrates computation, visualization, and programming in an easy to use environment where problems and solutions are expressed in familiar mathematical notation. Typical uses include:

•Math and computation
•Algorithm development
•Modeling, simulation, and prototyping
•Data analysis, exploration, and visualization
•Scientific and engineering graphics
•Application development, including graphical user interface building

Matlab is an interactive system whose basic data element is an array that does not require dimensioning. This allows to solve many technical computing problems, especially those with matrix and vector formulations, in a fraction of the time it would take to write a program in a scalar non interactive language such as C or Fortran. The name Matlab stands for matrix laboratory. Matlab was originally written to provide easy access to matrix software developed by the Linpack and Eispack projects. Today, Matlab uses software developed by the Lapack and Arpack projects, which together represent the state of the art in software for matrix computation. In university environments, it is the standard instructional tool for introductory and advanced courses in mathematics, engineering, and science. In industry, Matlab is the tool of choice for high productivity research, development, and analysis. Matlab features a family of application specific solutions called toolboxes.

b) MODELSIM

ModelSim is a multi language HDL simulation environment by Mentor Graphics for simulation of hardware description languages such as VHDL, Verilog and System C and includes a built in C debugger. Overview of the ModelSim simulation environment is divided into four: Basic simulation flow , Project flow ,Multiple library flow and Debugging tools.

c) ISE Design Suit

The ISE Design Suite: Embedded Edition includes Xilinx platform studio , software development kit large repository of plug and play IP including micro blaze and peripherals, and a complete RTL to bit stream design flow. Embedded Edition provides the fundamental tools, technologies and familiar design flow to achieve optimal design results.The ISE Design Suite: System Edition builds on top of the Embedded Edition by adding on System Generator for DSP™. System Generator for DSP is the industry's leading high-level tool for designing high-performance DSP systems using Xilinx all programmable devices, providing system modeling and automatic code

generation from Simulink and matlab ISE WebPACK delivers a complete, front-to-back design flow providing instant access to the ISE features and functionality at no cost. To learn more, please visit ISE WebPACK Design Software landing page.

d) DIGILENT ADEPT

The Digilent Communications Interface DLL, dpcutil.dll, provides a set of API functions for applications programs running on a Microsoft Windows based computer to exchange data with logic implemented in a Digilent system board. Various Digilent applications programs, such as Transport component of the Adept Suite, depend on the data interchange API functions in dpcutil. The operation of these functions depends on the presence of a Digilent communications subsystem component running the appropriate firmware and the implementation of the appropriate interface logic in the gate array. This include functional requirements of the communications interface logic and provides a sample implementation. The logic implements are set of registers in the gate array. An application on the host PC exchanges data with the design in the gate array by reading or writing these registers. Digilent Communications Interface modules implement the interface described in this document to control the reading and writing of these registers. Signals sourced by the host are generated by the Digilent communication interface and are inputs to the logic in the gate array. The term peripheral refers to the logic implemented in the gate array of the system board. Signals sourced by the peripheral are outputs from the logic implemented in the gate array.

2. Hardware requirements

a) Nexys3 Spartan 6

The Nexys3 is a complete, ready to use digital circuit development platform based on the Xilinx Spartan 6 LX16 FPGA. The Spartan 6 is optimized for high performance logic, and offers more than 50% higher capacity, higher performance, and more resources as compared to the Nexys2's Spartan-3 500E FPGA. Spartan 6 LX16 features include:

- GPIO includes 8 LEDs, 5 buttons,8 slide switches and 4-digit seven-segment display
- 2,278 slices each containing four 6 input LUTs and eight flip flops
- 576Kbits of fast block RAM
- two clock tiles (four DCMs & two PLLs)
- 32 DSP slices
- 500MHz+ clock speeds
- Xilinx Spartan 6 LX16 FPGA in a 324 pin BGA package
- 16Mbyte Cellular RAM (x16)
- 16Mbytes SPI (quad mode) PCM non volatile memory
- 16Mbytes parallel PCM non-volatile memory
- 10/100 Ethernet PHY
- On-board USB2 port for programming & data xfer
- USB-UART and USB HID port (for mouse/keyboard)
- 8 bit VGA port
- 100MHz CMOS oscillator
- 72 I/O's routed to expansion connectors
- USB2 programming cable included

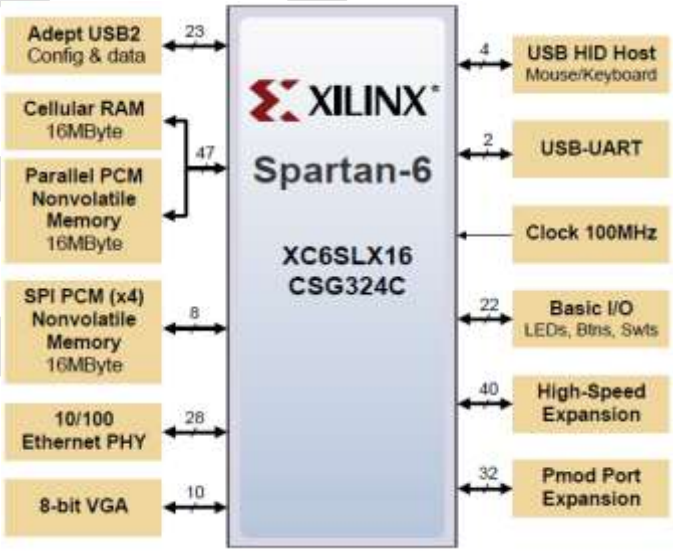

Figure 4. Nexys3 spartan 6

b) Memory Card

A memory card or flash card is an electronic flash memory data storage device used for storing digital information. These are commonly used in many electronic devices, including digital cameras, mobile phones, laptop computers, MP3 players and video game consoles, Tablets. Most of these can be diminutive, re-recordable, and can retain data without power.

c) USB cable

The Universal Serial Bus (USB) is an industry standard developed in the mid 1990s that defines the cables, connectors and communications protocols used in a bus for connection, communication, and power supply between computers and electronic devices.

ACKNOWLEDGMENT

First I would like to thank our god for giving me the strength to finish this work; To my family especially to my father ,mother and cousins for their moral and financial support in order to finish this work; To all my friends and classmates who had helped to do this study presentable.

CONCLUSION

Reversible data hiding in encrypted images is a advanced topic provides the privacy preserving requirements from cloud data management. Reserving room before encryption shows the excellent property that the original cover is recovered without any loss after the embedded data is extracted out while protecting the image content's confidentiality. This method can take advantage of all traditional RDH techniques for plain images and achieve excellent performance without loss of perfect secrecy.. All previous methods embed data by reversibly vacating room from the encrypted images, which subject to some errors on data extraction and image restoration. Reversible data hiding is a novel method by reserving room before encryption with a traditional Reversible data hiding algorithm, and thus it is easy for the data hider to reversibly embed data in the encrypted image. Also it achieve real reversibility, that is data extraction and image recovery are free of any error. This novel Method achieve real reversibility by the use of Rationale Rhombus algorithm, and provide great improvement on the quality of marked decrypted images while retrieving the image.

REFERENCES:

[1] Kede Ma, Weiming Zhang, Xianfeng Zhao, Member, IEEE, Nenghai Yu, and Fenghua Li "Reversible Data Hiding in Encrypted Images by Reserving Room Before Encryption".

[2] Vasiliy Sachnev, Hyoung Joong Kim, Member, IEEE, Jeho Nam Senior Member, IEEE,Sundaram Suresh, and Yun Qing Shi, Fellow, IEEE "Reversible Watermarking Algorithm Using Sorting and Prediction.

[3] Lixin Luo, Zhenyong Chen, Ming Chen, Xiao Zeng, and Zhang Xiong "Reversible Image Watermarking Using Interpolation Technique".

[4] Mona M. El-Ghoneimy, Associate Professor, Elect. & Comm. Dept.,Faculty of Engineering, Cairo University, Post code 12316"Comparison Between Two Watermarking Algorithms Using Dct Coefficient, And Lsb Replacement.

[5] Deepshikha Chopra, Preeti Gupta, Gaur Sanjay B.C, Anil Gupta "Lsb Based Digital Image Watermarking For Gray Scale Image".

[6] Masoud Nosrati Ronak Karimi Mehdi Hariri "Reversible Data Hiding: Principles, Techniques, and Recent Studies".

[7] Yun Q. Shi, Department of Electrical and Computer Engineering,New Jersey Institute of Technology, Newark, NJ 07102, USA0"Reversible DataHiding".

[8] W. Zhang, B. Chen, and N. Yu, "Improving various reversible data hiding schemes via optimal codes for binary covers," IEEE Trans. Image Process., vol. 21, no. 6, pp. 2991–3003, Jun. 2012.

[9] J. Fridrich and M. Goljan, "Lossless data embedding for all image formats," in Proc. SPIE Proc. Photonics West, Electronic Imaging, Security and Watermarking of Multimedia Contents, San Jose, CA, USA, Jan. 2002, vol. 4675, pp. 572–583.

[10] M. Johnson, P. Ishwar, V. M. Prabhakaran, D. Schonberg, and K. Ramchandran, "On compressing encrypted data," IEEE Trans. Signal Process., vol. 52, no. 10, pp. 2992–3006, Oct. 2004.

[11] M. Goljan, J. Fridrich, and R. Du, "Distortion-free data embedding for images," in Proc. Inform. Hiding Workshop, Pittsburgh, PA, 2001, pp. 27–41.

[12] R.Gonzalez and R.Woods, "Digital Image Processing", 1998.

[13] B Surekha, Dr GN Swamy, "A Spatial Domain Public Image Watermarking", International Journal of Security and Its Applications Vol. 5 No. 1, January, 2011.

[14] W. Zeng, "Digital watermarking and data hiding: technologies and applications," in Proc. Int. Conf. Inf. Syst., Anal. Synth., vol. 3, 1998, pp. 223–229.