National Conference on Recent Innovations in Computer Science & Communication Engineering, *July 2016* *ISBN: 978-93-86005-02-1*

Image Hiding Application using Embedded Extended Visual Cryptography on Secret Natural Colour Image by Additive Colour-Space Distribution

Harinandan Tunga*, Stainlee J. Bakhla, Debadeep Basu, Diptanil Das and Satyaki Siddhanta

Department of CSE, RCCIIT, Kolkata - 700015, West Bengal, India; harinandan.tunga@gmail.com

Abstract

Extended Embedded Visual Cryptography widens the concept of Visual Cryptography, by hiding or embedding the generated shares into host images, such that the original image may be reconstructed from these host images but by themselves, the images do not reflect the presence of the secret image thus making it in-detectable.In this paper, we intend to take up the challenge of Colour Image Cryptography, and implement it to find the most efficient algorithm or process for Visual Cryptography by comparing various existing cryptographic schemes and also some recent developments in the field of colour image cryptography. The secret image to be encrypted is first split into its three component shares based on the additive colour scheme (RGB) and then they are embedded into cover images so as to successfully hide them from unwanted viewers. The host images or cover images to be used are selected from a set of images in a gallery by comparing the respective colour space of the image with the colour space of the available images to be used as host. The images which provide the least distortion are selected to be encoded with the secret image data, and the host images are replaced with the embedded images in the gallery, thereby successfully hiding the secret image among the other images. Decryption is achieved by providing the hidden host images as key and reconstructing the secret image.

Keywords: Color Image, Color Space, Embedded, Extended, Image Hiding, Visual Cryptography

1. Introduction

Visual cryptography is a popular solution for image encryption. Using secret sharing concepts, the encryption procedure encrypts a secret image into the shares (printed on transparencies) which are noise-like secure images which can be transmitted or distributed over an untrusted communication channel. Using the properties of the Human Visual System (HVS)[2] to force the recognition of a secret message from overlapping shares, the secret image is decrypted without additional computations and any knowledge of cryptography.

**Author for correspondence*

Visual cryptography was proposed by[1] who introduced a simple but perfectly secure way that allows secret sharing without any cryptographic computation[3], which they termed as Visual Cryptography Scheme (VCS). The simplest Visual Cryptography Scheme is given by the idea of A secret image consists of a collection of black and white pixels where each pixel is treated independently.

In this paper, we intend to take up the challenge of Colour Image Cryptography, and implement it to find the most efficient algorithm or process for Visual Cryptography by comparing various existing cryptographic schemes[5,7,8] and also some recent developments in the field of colour image cryptography.

In this paper, we have various sections. Section 2 deals with the Reviewing of Literature available on Visual Cryptography and Embedded Extended Visual Cryptography. Section 3 contains the Problem Statement or Objective of the paper. Section 4 consists of a System design that is the software that we wish to implement. Section 5 and 6 consists of the Formualtion and Short Description of the process involved in the paper. Section 7 contains some images

2. Review of Literature

Visual cryptography was proposed by[1] who introduced a simple but perfectly secure way that allows secret sharing without any cryptographic computation, which they termed as Visual Cryptography Scheme (VCS). The simplest Visual Cryptography Scheme is given by the idea of A secret image consists of a collection of black and white pixels where each pixel is treated independently.

In the example depicted in Figure 1, the image containing the Wikipedia logo, has been split into two component images. Each component image has a *pair* of pixels for every pixel in the original image. These pixel pairs are shaded black or white according to the following rule: if the original image pixel was black, the pixel pairs in the component images must be complementary; randomly shade one ▪□, and the other □▪. When these complementary pairs are overlapped, they will appear dark gray. On the other hand, if the original image pixel was white, the pixel pairs in the component images must match: Both ▪□ or both □▪. When these matching pairs are overlapped, they will appear light gray.

So, when the two component images are superimposed, the original image appears. However, considered by itself, a component image reveals no information about the original image; it is indistinguishable from a random pattern of ▪□/□▪ pairs. Moreover, if you have one component image, you can use the shading rules above to produce a *counterfeit* component image that combines with it to produce any image at all.

Since the initiation visual cryptography has seen many reforms and various cryptographic schemes have been developed that have turned meaningless images being shared to meaningful images that contain hidden information.

Liu and Wu[4] devised a method of image cryptography using meaningful shares containing embedded secret image information. Liu and Wu

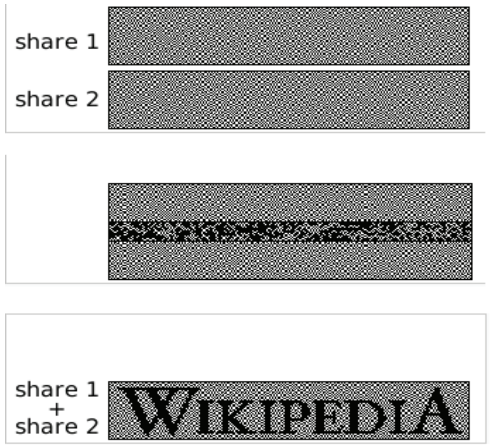


Figure 1. Simple example of traditional visual cryptography using the wikipedia logo.

made use of Grey scale images that had been half toned to binary images, i.e., each pixel is either a black pixel or a white pixel. In the process outlined by Liu and Wu, they followed traditional VCS technique for share generation where each image pixel is broken in 2 or more sub-pixels such that if they are overlaid, the original image is reproduced. They suggested that if m shares are made of an image, they can be embedded in any m out of n number of cover images. such that stacking a correct combination of those m images will result in the decryption of the secret image.

Thus, in Embedded Cryptography, an image I_0, say, is broken into m shares such that each pixel has m sub-pixels and they are embedded into m out of n shares.

Jerripothula Sandeep and Abdul Majeed in their paper6 on 'Extended Embedded Visual Cryptography' published in IOSR Journal of Computer Engineering (IOSRJCE), describe the process of Embedded VCS as shown in Figure 2.

Further reviewing of many recent published papers led to the realisation of a problem that is discussed in the following sections.

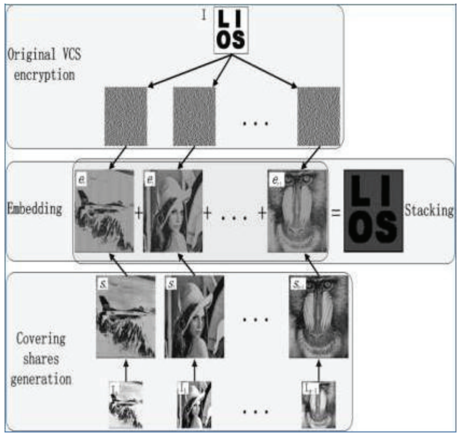


Figure 2. Traditional embedded extended visual cryptography process.

3. Objective

After going through various papers and journals on Embedded Extended Visual Cryptography, it is seen that after embedding of n-parts of an Image onto n-covering shares, there is significant amount of distortion in the cover images. When passing these images, the distortions in the images may lead to suspicion of observer that they may contain hidden information. Also it is seen that most processes have made use of grayscale images, and very few have been used for the implementation of color images.

Also, in today's world almost all transmissions take place over the Internet and everything is done on the computer, and image sharing can be made secure by providing a safe and simple Encrypting and Decryption Interface, one which is quite unavailable in the current situation. Also, our paper aims to do a comparative study on the existing methods of color image cryptography and implement embedded sharing by creating shares based on the basic color matrices of an image instead of splitting a pixel.

4. System Design

The Encryption-Decryption Interface shall be based on a standard file uploading UI, and shall be written in Java code. Further implementation can lead to webpage design where a user can store his secret on a server and review the secret image whenever needed.

The aim is to design two interfaces – An Encryption interface, where a user wishing to encrypt an image will upload a secret image, and 3 additional images he/she wishes to set as cover images. The system shall embed the images with respective shares and store them for use or for transport, and A Decryption Interface where a user can enter his/her cover images and he/she shall obtain the secret image. The Context Level Diagram in Figure 3 outlines the proposed System Overview

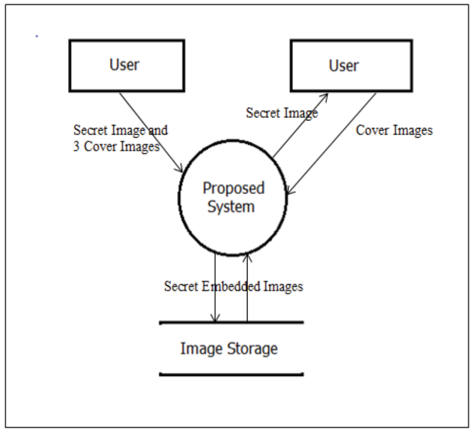


Figure 3. Context level diagram of proposed system.

In the following sections, we shall discuss about the proposed method of image encryption and decryption.

5. Methodoly of Implementation

Implementation of encryption process is achieved by the following steps

Step 1: Accept Secret Image as Input from user.
Step 2: Accept 3 Cover Images as Input from user.
Step 3: Split the Secret Image into 3 shares corresponding to its subsequent Color Space matrix in R,G and B format.
Step 4: Choose the proper Cover image for each colour space based on the minimum difference between the corresponding colour matrix and the colour matrix of the cover image.
Step 5: Embed ά amount of pixel information into each pixel of the specific corresponding color space, where ά ranges from 0 to 1 by bitwise logical XOR between the cover image pixel and color space pixel.
Step 6: Repeat Step 5 till all pixels have been modified or checked.
Step 7: Repeat Steps 4-6 for all 3 Colour spaces.

Decryption Process is a reverse implementation requiring the original cover images.

Step 1: Accept original cover images as input from user.
Step 2: Select respective share image based on least pixel difference.
Step 3: Perform bitwise XOR between original cover and embedded cover to generate a single color space equivalent of secret image on each pixel.
Step 4: Repeat step 3 till we get a dark result image.
Step 5: Multiply the pixel matrix of the obtained image by 1/ά to obtain the original secret image pixel intensity for that colour space.
Step 6: Perform steps 3 and 4 till all 3 colour space images are generated.
Step 7: Stack the obtained images to get the original secret image.

5.1 Discussion on RGB Image

An RGB image, sometimes referred to as a *truecolor* image, is stored as an *m*-by-*n*-by-3 data array that defines red, green, and blue color components for each individual pixel. RGB images do not use a palette. The color of each pixel is determined by the combination of the red, green, and blue intensities stored in each color plane at the pixel's location. Graphics file formats store RGB images as 24-bit images, where the red, green, and blue components are 8 bits each. This yields a potential of 16 million colors. The precision with which a real-life image can be replicated has led to the nickname "truecolor image".

RGB Images are stored in memory as an overlapped map of 3, 2 dimensional matrices. An RGB image with m-by-n size contains mn data in each two dimensional matrix. Each cell of each m-by-n matrix denotes a single pixel in the image of that colour space (for example the pixel value for Red at position P_{ij} can be 201). Each pixel in a colour space has a maximum size of 8 bits. That is, each pixel can have a maximum of 256 values (0–255). Let us represent this following image (Figure 4.) as a matrix of red green and blue.

The given image is a 7 x 7 image containing 7 colours of the visible spectrum. For visibility, it has been scaled 20 times.

The corresponding matrices for Red, Green and Blue Components are shown below:

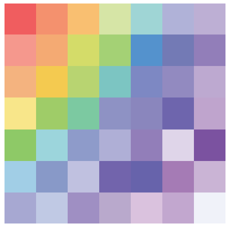


Figure 4. 7 x 7 RGB image.

RED

$$\begin{vmatrix} 255 & 254 & 255 & 240 & 123 & 156 & 164 \\ 255 & 255 & 234 & 157 & 153 & 144 & 188 \\ 254 & 215 & 171 & 126 & 148 & 177 & 177 \\ 208 & 154 & 126 & 139 & 157 & 146 & 193 \\ 156 & 96 & 121 & 152 & 164 & 139 & 219 \\ 169 & 137 & 160 & 154 & 208 & 196 & 216 \\ 195 & 167 & 195 & 198 & 158 & 215 & 245 \end{vmatrix}$$

GREEN

$$\begin{vmatrix} 95 & 150 & 184 & 233 & 251 & 205 & 174 \\ 141 & 174 & 202 & 240 & 210 & 169 & 193 \\ 192 & 219 & 237 & 216 & 172 & 183 & 152 \\ 240 & 238 & 198 & 158 & 160 & 113 & 168 \\ 210 & 156 & 140 & 133 & 138 & 95 & 185 \\ 181 & 130 & 133 & 117 & 180 & 135 & 172 \\ 172 & 134 & 167 & 150 & 55 & 175 & 239 \end{vmatrix}$$

BLUE

$$\begin{vmatrix} 94 & 138 & 128 & 132 & 128 & 219 & 255 \\ 109 & 115 & 84 & 118 & 212 & 248 & 255 \\ 111 & 102 & 113 & 171 & 251 & 255 & 255 \\ 168 & 142 & 194 & 250 & 255 & 255 & 255 \\ 210 & 217 & 250 & 255 & 255 & 253 & 254 \\ 253 & 254 & 254 & 254 & 255 & 255 & 255 \\ 255 & 255 & 255 & 254 & 255 & 252 & 253 \end{vmatrix}$$

Thus we see that the colour image is broken into 3, p x q matrices which represent each of the colour spaces. The colour of a pixel at any point is obtained by an overlapping combination of the three pixels at each pixel position. For example, a Black Pixel is denoted by (0,0,0) and White Pixel is denoted by (255,255,255)

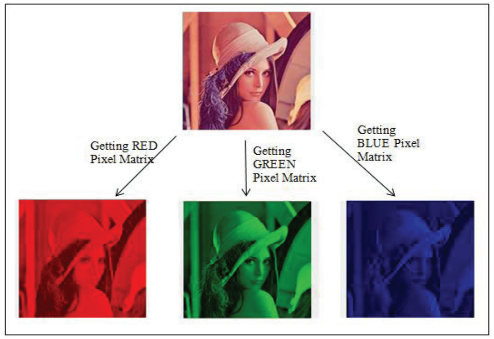
5.1.1 *Input*

The User enters an RGB image I_{secret} which he wants to secure, and three other images that shall function as cover images (say C_R, $C_{G,}$ and C_B). The reason for naming the covers as we have shall be discussed in detail in the following sub-section.

5.1.2 *Generation of Share Images*

Unlike traditional VCS which works to split a half-toned image pixel by pixel into m sub-pixels for embedding, this process involves separation of the secret image into its corresponding constituent colour spaces based on the additive colour model. Choosing the additive colour model, over the subtractive colour model is solely done because this system aims to be restricted to the computers. Over the internet and network, the additive model is preferred as almost all Computer monitors display colour using the additive RGB model, where no pulse (0 value) is depicted as an empty pixel (or black pixel), unlike the CMY model whose 0 value corresponds to white.

In this system, we simplify the share generation schema of Visual Cryptography by splitting the colour image into its corresponding color matrices. As shown in Figure 4, the colour image contains three m x n matrices where m-by-n is the image width-by-height. In the figure below (Figure 5), we can see how the image of Lena, is split into 3 shares – i.e., its corresponding Red, Green and Blue colour spaces, where R,G,B ∑ [0,255].



Getting BLUE Pixel Matrix

Figure 5. Splitting into colour components.

5.1.3 Selecting the Cover Images

The user also enters cover images along with the secret image. The optimal cover for each image is selected by checking the minimum difference between corresponding pixels between the images and the shares generated by splitting the 3 x m x n matrix into 3 individual color space matrices. For the above image let's say we take the following cover images.

5.1.4 Embedding Process

The Embedding Process is a fairly simple operation. In this process, three shares are generated by embedding the individual colour space pixel values onto the cover image pixel values for that corresponding color space. Embedding is done by the following algorithm

Step 1: Select a pixel P_{ij} on the secret image colour space matrix (say Red).

Step 2: Select a pixel P_{xy} on the cover image colour space matrix for that colour.

Step 3: Perform the following operation

$$P_{(x,\,y)\ \text{Result Red}} = (1-\alpha)\mathrm{P}_{xy\ Cover\ Red}\ \ XOR\ \ \alpha(P_{xy\ Secret\ Red})$$

Where $\alpha \in (0, 1)$.

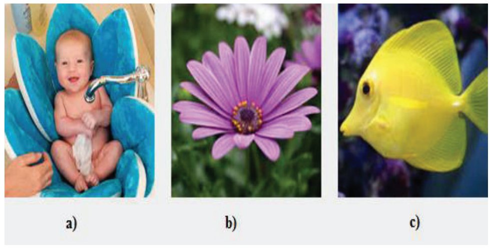


Figure 6. Covering images.

Here α is the amount of pixel data that is being embedded from the secret image into the cover image. It is used to check the specific limit where Quality of Resultant Image and Quality of Embedded Shares is at a reasonable value. Given by the above relation, it is quite easy to note that they are inversely proportional to each other, i.e., if the value of α is decreased towards 0, the quality of the embedded shares increases, but the quality of the resultant image decreases. Although, to HVS, it is hardly perceivable in small amounts.

The XOR Operation mentioned in the equation performs a bitwise XOR of the binary values of the pixels. We shall demonstrate the XOR operation on the next page.

Below we have the truth table for a single bit XOR operation

We see that for similar bits the output is 0, and for unlike bits the output is 1. Let us demonstrate the XOR operation on image pixels now Let us say, we are viewing a part of the image as a 2 x 2 square, where each cell holds a pixel value for that colour space.

Say, the matrix $\begin{pmatrix} 237 & 201 \\ 192 & 83 \end{pmatrix}$ is a 2 x 2 portion in the secret image and the matrix.

$\begin{pmatrix} 221 & 35 \\ 72 & 113 \end{pmatrix}$ is a 2 x 2 portion in the cover image. Now, we examine the matrix elements as individual pixel values. Taking the pixel value at (0,0) for each and XOR using the formula mentioned before we have,

A	B	RESULT
0	0	0
0	1	1
1	0	1
1	1	0

$(1-\alpha)$ *Pixel of cover image XOR*

α . *Pixel of Secret Image*

Let us say for these set of images, the value of $\alpha = 0.25$, thus for the pixel at (0,0) we see

$$\begin{aligned} & 0.75 * 221 \ \ XOR \ \ 0.25 * 237 \\ = \ & 165 \ \ XOR \ \ 59 \\ & (\textit{rounding off the Floor value}) \\ = \ & 10100101_2 \quad XOR \ \ 00111011_2 \\ & (\text{converting to equivalent binary}) \\ = \ & 10011110_2 \\ & (\text{resultant } XOR) \\ & = 158_{10} \quad (\text{converting back to decimal}) \end{aligned}$$

Thus, the pixel value for the embedded share image is 158 for the pixel location (0,0).

Similarly performing XOR on the other pixels, we get the result matrix

$$\begin{pmatrix} 158 & 40 \\ 6 & 64 \end{pmatrix}$$

Using this logic, all pixels in the cover image are embedded with a certain amount of data of the secret image.

Step 4: Select next pixel.

Step 5: Repeat Steps 1-4 till whole matrix is embedded for the corresponding covering share.

Step 6: Take next cover image and take its Green Colour Space and using Secret Share Green, repeat Steps 1-5.

Step 7: Repeat the above Steps for Blue Colour Space of 3rd cover image and Blue Secret Share.

Thus it is quite evident why we had decided to name the covering shares as C_R, C_G and C_B. It is because those cover images consist the corresponding embedded colour share. C_R contains Red. C_B contains Blue and C_G contains Green. It is also important to note that although the embedded covers have been altered in their respective matrices, they are still complete RGB images as they have all 3 colour spaces (2 same as original cover and 1 embedded with share image colour matrix) and are meaningful as standalone images.

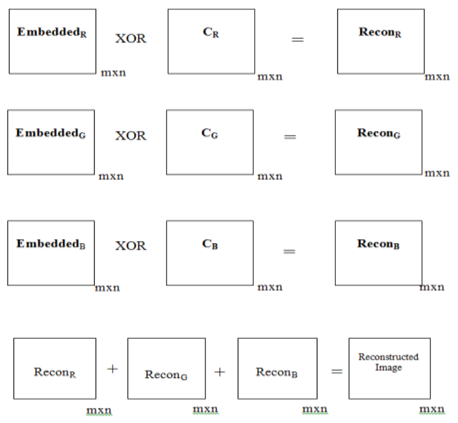
6. Regeneration of Original Secret Image

Original Secret Image is generated by performing pixel wise bitwise XOR operation on the original cover image and the embedded share image, to produce an individual colour space reconstructed image. On overlapping these colour space reconstructions, we return back the original secret image.

7. Sample Testing

At the present scenario, a very basic embedding and extracting console program in Java has been implemented, through which we can split a secret image into consequent share images of their respective color spaces, and embed them into the cover images and using the original cover images, re-obtain the original secret image.

The following screenshots are taken of testing of embedding and extraction process with the following images.


Embedded$_R$ XOR C$_R$ = Recon$_R$
mxn
Embedded$_G$ XOR C$_G$ = Recon$_G$
mxn
Embedded$_B$ XOR C$_B$ = Recon$_B$
mxn
Recon$_R$ + Recon$_G$ + Recon$_B$ = Reconstructed Image

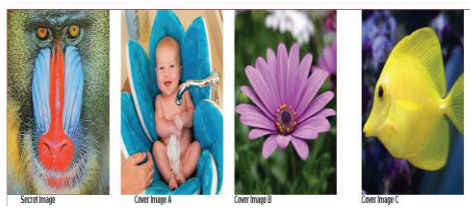


Figure 7. From left (secret image, cover image A, cover image B, cover image C).

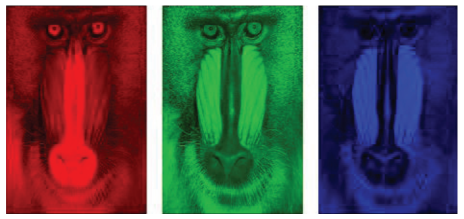


Figure 8. Components of secret image from left (red component, green component, blue component).

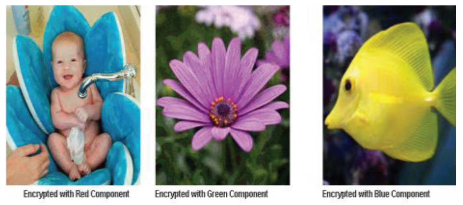

Encrypted with Red Component
Encrypted with Green Component
Encrypted with Blue Component

Figure 9. Cover images after embedding color components at α=0.1. From left (encrypted with red component, encrypted with green component, encrypted with blue component).

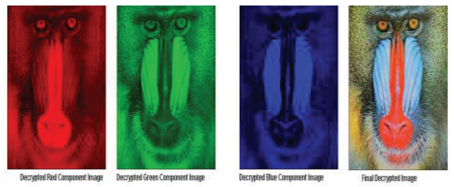

Decrypted Red Component Image
Decrypted Green Component Image
Decrypted Blue Component Image
Final Decrypted Image

Figure 10. color component images and final superimposed image.

8. Scope of Future Work

At the present time, we have been able to split an image into its consequent RGB spaces and embed them into cover images using XOR operator with a certain percentage of the pixel information of the secret image, and reconstruct the original image using the covering shares and embedded shares.

Future Work is to implement the above algorithm from console based application with a functional Graphical User Interface, and to reduce the use of original cover images as keys in Decryption, and also implement a form of scrambling of keys to further protect the data.

9. References

1. Naor M, Shamir A. Visual cryptography. Eurocrypt. 1994.

2. Kaur N, Mahajan R. An improved scheme of extended embedded visual cryptography. International Journal of Computer Engineering and Science. 2014 Mar; 8(1):41–7.
3. Patil S, Rao J. Extended visual cryptography for color shares using random number generators. 2012; 1(6):399–412
4. Liu F, Wu CK. Embedded extended visual cryptography schemes. IEEE Transactions on Information Forensics and Security. 2011; 6(2):207–22.
5. Verma J, Khemchandani V. A visual cryptographic technique to secure image shares. IJERA.2012; 2(1):1121–5.
6. Sandeep J, Majeed A. Embedded extended visual cryptography scheme. IOSRJCE. 2012; 8(1):41–7.
7. Tunga H. A new secret coloured image encryption and decryption scheme based on (2, 2) Visual Cryptography Scheme (VCS). International Journal of Computer Applications. 2014 Sep; 101(12):13–5.
8. Abdulla S. New visual cryptography algorithm for colored image. Journal of Computing. 2010 Apr; 2(4):21–5.