symmetry

MDPI

Article

Flip Extended Visual Cryptography for Gray-Scale and Color Cover Images

Lu Wang [1], Bin Yan [1], Hong-Mei Yang [2,*] and Jeng-Shyang Pan [2]

[1] College of Electronic and Information Engineering, Shandong University of Science and Technology, Qingdao 266590, China; wanglu26@hotmail.com (L.W.); yanbinhit@sdust.edu.cn (B.Y.)
[2] College of Computer Science and Engineering, Shandong University of Science and Technology, Qingdao 266590, China; jspan@cc.kuas.edu.tw
* Correspondence: skd991737@sdust.edu.cn

Abstract: Visual cryptography (VC) has found numerous applications in privacy protection, online transaction security, and voting security, etc. To counteract potential cheating attacks, Lin et al. proposed flip visual cryptography in 2010, where a second secret image can be revealed by stacking one share with a flipped version of another share. The second secret image can be designed as an additional verification mechanism. However, Lin's scheme produces meaningless shares and is only applicable to binary secret images. It is interesting to explore whether it is possible to extend the flip VC to having cover images (i.e., extended VC) and these cover images are color images. This problem is challenging since too many restricting conditions need to be met. In this paper, we designed a flip VC for gray-scale and color cover images based on constraint error diffusion. We show that it is possible to meet all the constraints simultaneously. Compared with existing schemes, our scheme enjoys the following features: Color cover images, no computation needed for decoding, and no interference from cover image on the recovered secret image.

Keywords: flip visual cryptography (FVC); meaningful shares; error diffusion halftoning

Citation: Wang, L.; Yan, B.; Yang, H.-M.; Pan, J.-S. Flip Extended Visual Cryptography for Gray-Scale and Color Cover Images. *Symmetry* **2021**, *13*, 65. https://doi.org/10.3390/sym13010065

Received: 12 November 2020
Accepted: 28 December 2020
Published: 31 December 2020

1. Introduction

In today's world, information is a popular application resource and an essential carrier of communication in our lives. The rapid development of science and technology has driven the continuous advancement of information technology, and more people have begun to pay attention to the importance of information. At the same time, more and more people have put forward more security requirements for the protection of their privacy [1–3]. Visual cryptography (VC) is an important branch of information security and has been further developed in the field of secret sharing. It can not only ensure the security of people's privacy but also has a simple decoding operation and numerous potential applications. Therefore, visual cryptography technology has become a hot research direction since its proposal in 1994.

Naor and Shamir designed the first visual cryptography algorithm in 1994 [4]. The basic idea is to replace each pixel in an image with a block of pixels. The pixel information in the pixel block is filled with the superposition between pixels, and finally, an encrypted image cannot be recognized by the human eye [4–13]. For a (k, k)-threshold scheme, the information transmitting end encrypts the secret image into k share images and transmits them to the receiving end. After receiving the k share images, the receiving end superimposes and restores the secret image. The recovered secret image is displayed in black and white by the synthesized pixels. If the receiving end receives less than k share images, the secret image cannot be recovered. Therefore, the visual cryptography scheme has sufficient security and feasibility. However, if a secret image is encrypted by this scheme, the pixels are expanded to m pixels, in order to generate k noise-like share images. The size of the recovered secret image is m times of the size of the original image. Each of the black

pixels and the white pixels is composed of a certain number of sub-pixels, i.e., a mixture of black and white. So, the visual quality of the recovered secret image is not ideal. Therefore, more visual cryptography algorithms tried to improve the image quality of the stacked image [14,15].

There is a trade-off between the quality of the recovered secret image and the size expansion. Ordinary visual cryptography divides the secret image pixels into blocks and embeds the secret pixel information with the basic matrix. Since each secret pixel can be rendered by a block of pixels in a recovered image, ordinary visual cryptography has better visual quality. However, the recovered secret image is larger than the original secret image [5]. In 2002, Nakajima and Yamaguchi proposed extended visual cryptography for natural images [16]. This scheme can produce a meaningful share of images. The use of halftone technology and contrast-enhancing methods improves the quality of recovering secret images. In 2011, Liu and Wu proposed a scheme that combined extended VC with digital halftoning. Embedding a cover image into a share image improves the quality of the share image [17].

The development of meaningful visual cryptography provides more room for the development of subsequent visual cryptography technologies because, with meaningful VC, share images can be stored in large quantities, which is convenient for encrypting more secret information. Naor and Shamir also designed the first meaningful VC [4], which has a good visual quality and is easier to store multiple share images. At the same time, share images are less likely to attract the attention of attackers. Image encryption often uses halftone images and gray-scale images, so meaningful visual cryptography requires high fidelity for share images. There are mainly two types of meaningful visual cryptography algorithms: the first is to process in the frequency domain, and then use the original encryption algorithm to embed the encrypted secret image in the frequency domain of the cover image by wavelet transform, and finally output a gray band share images with covered images [18]. For the second method, the gray-scale image is halftoned in the spatial domain, and then the cover image is embedded in the share image using the pixel compensation method [19].

Color visual cryptography is also one of the main researches in the field of visual cryptography [20,21]. Color visual cryptography has good visual quality. The earliest research on color visual cryptography was the color visual cryptography (k, n) solution. The secret image is encrypted into n share images, and no less than k share images are taken to superimpose to obtain the secret image. In 2000, Yang and Laih constructed a new color visual cryptography scheme. The basic structure of the sub-pixels can be directly used for the proposed structure in the image editing package, and it is easy to implement [6]. In 2003, Hou proposed VC algorithms for color images. He proposed three VC methods for VC and grayscale images. These methods are based on VC for binary images, digital halftoning, and color decomposition methods. These methods have similar VC advantages to binary images [22]. In 2011, Prakash and Govindaraju proposed a color visual cryptography scheme based on direct binary search (DBS) with adaptive searching and exchange functions [23]. Through this solution, a better halftone image can be generated, in addition to lossless recovery.

We conclude the literature review by noting that, even though a color extended VC with authentication capacity is desired, currently no such design is reported. One possible reason is that to design such a VC system, one needs to meet many restrictions.

In this paper, a meaningful flipping visual cryptography algorithm is proposed based on the constraint error diffusion. The main features of our system are as follows:

1. The cover image can be a gray-scale image or color image. So, the share image is meaningful.
2. The share image and the cover image are of the same size, and there is no pixel expansion after encryption.

3. Two secret images are simultaneously embedded. That is, by superimposing two share images on the front side, one obtain a secret image $\boldsymbol{I}_1$; by superimposing the share image $\boldsymbol{S}_1$ and the flipped share image $\boldsymbol{S}_2$, one obtain the second secret image $\boldsymbol{I}_2$.
4. When the secret image $\boldsymbol{I}_2$ are recovered by stacking, there is no interference from the cover image.

To the best of our knowledge, currently, there is no VC algorithm that meets all the above requirements.

The structure of this paper is as follows: Related background is reviewed in Section 2. In Section 3, we present a meaningful flipping visual cryptography algorithm. Section 4 combines the color VC algorithm to obtain color meaningful flip VC. Section 5 provides the experimental results. Section 6 concludes this paper.

2. Related Works

This paper is divided into two parts: gray-scale meaningful flipping visual cryptography and color meaningful visual cryptography. Meaningful flipping visual cryptography uses the method of constrained error diffusion to embed the cover image into the share images, which improves the visual quality of the share images and can better protect the secret image from being discovered. Color flip visual cryptography can encrypt color secret images, and the share images are color images with better visual quality. In this section, we review the basic structure of VC with meaningful shares and flip VC [24,25]. Unless otherwise stated, all the secret images in this section and next section are binary images.

We would like to ask our readers to pay attention to the difference between the secret image and the share images.

1. A binary secret image is a digital image stored in a computer disk and/or memory, and it consists of white pixels and black pixels. Following the convention in VC and digital halftoning, we use a '1' to represent a black pixel on the secret image, and we use a '0' to represent a white pixel on the secret image.
2. A share image is printed on a transparency. When printing on paper, which is white by default, a printer only needs to print a black dot or print 'no dot'. Most printers are not able to print white color except for some UV printers that are designed to print on cloth and plastic. So, when printing a share image on a transparency, a black pixel is printed as black while a white pixel is printed as 'transparent' since no dot is printed. So, when referring to a 'white' pixel on a transparency, we use the term 'transparent' pixel. Following the convention in VC and digital halftoning, we use a '1' to represent a black pixel on a transparency, and we use a '0' to represent a transparent pixel on a transparency.

2.1. VC with Meaningful Shares

A VC system with meaningful shares is also referred to as the extended VC in the literature. The overall structure of such a system is illustrated in Figure 1. For illustration purposes, we consider the case with two secret images and two cover images. A secret image is an image that the holder would like to share and would like to keep confidential during the sharing process. It can be a binary image, grayscale image, or even color image. In Figure 1, $\boldsymbol{I}_1$ and $\boldsymbol{I}_2$ are two secret images. In an ordinary VC, these secret images are encoded into share images $\boldsymbol{S}_1$ and $\boldsymbol{S}_2$ that are noise-like (i.e., meaningless). These share images are printed on transparencies, such as projector films or other transparent plastic thin sheets. Then the share images are distributed to different parties (also called participants). However, these noise-like shares may arouse the suspicion of an attacker. Extended VC is a countermeasure to this issue, where natural-looking cover images are used to generate shares. The two images $\boldsymbol{C}_1$ and $\boldsymbol{C}_2$ are two cover images. After gathering enough shares, a VC decoder can decode the secret images by simply stacking the transparencies carrying the shares. By inspecting the stacking results, a human is able to read the secret using his/her human vision system. So a VC decoder, in its strict sense, doesn't need computation. In Figure 1, $\hat{\boldsymbol{I}}_1$ and $\hat{\boldsymbol{I}}_2$ are two decoded secret images. These

images are also referred to as target images in the VC literature. The target image is usually not the same as its secret image counterpart. However, as long as the content of the secret image can be recognized from the target image, one deems the decoding as valid.

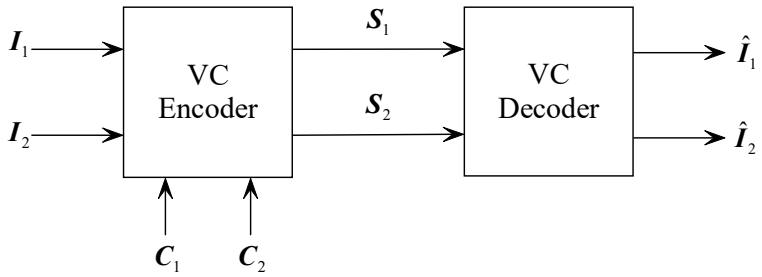


Figure 1. Structure of a VC system with meaningful shares.

The most basic and simple algorithm model of visual cryptography is the (2, 2) scheme proposed by Naor and Shamir, which can be considered as a degenerated version of extended VC. It is also a building block of extended VC, so we review it briefly with a focus on its encoder. In the (2, 2)-threshold scheme, the secret image is encrypted into two share images. Even if an attacker obtains one of the share images, the secret image cannot be inferred from it. Two share images are needed in order to recover the secret image. The (k, n) threshold VC is an extension of (2,2) threshold scheme, that is, the secret image is encrypted to generate n share images. Only when k or more share images are superimposed, the secret image can be restored. A $(2,2)$-threshold scheme consists of two collections of matrices, such as:

$$\mathcal{C}_0 = \left\{ \begin{pmatrix} 1 & 0 \\ 1 & 0 \end{pmatrix}, \begin{pmatrix} 0 & 1 \\ 0 & 1 \end{pmatrix} \right\}, \mathcal{C}_1 = \left\{ \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \right\} \tag{1}$$

It should be noted that all the matrices in a set $\mathcal{C}_i$, where $i = 0, 1$, can be obtained by column permutation of a basic matrix. For example, all matrices in $\mathcal{C}_0$ can be obtained by column permutation of the matrix

$$\boldsymbol{A}_0 = \begin{pmatrix} 1 & 0 \\ 1 & 0 \end{pmatrix}. \tag{2}$$

Similarly, all matrices in $\mathcal{C}_1$ can be obtained by column permutation of the matrix

$$\boldsymbol{A}_1 = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}. \tag{3}$$

So $\boldsymbol{A}_0$ and $\boldsymbol{A}_1$ are usually called the basic matrices for this $(2,2)$-threshold VC scheme [24].

To encode a white pixel, one randomly chooses a matrix $\boldsymbol{C} \in \mathcal{C}_0$ from the set $\mathcal{C}_0$ and assign the first row to share 1 and the second row to share 2. Likewise, when encoding a black pixel, one randomly chooses a matrix $\boldsymbol{D} \in \mathcal{C}_1$ from the set $\mathcal{C}_1$ and assign the first row to share 1 and the second row to share 2.

To illustrate this encoding process, let us consider a toy example as shown in Figure 2. The secret image consists of two pixels on a white paper, a black pixel (top) and a white pixel (bottom). To encode the black pixel, we randomly choose a matrix from the set $\mathcal{C}_1$. Suppose that the matrix

$$\boldsymbol{D} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \tag{4}$$

is chosen. Then the first row of $\boldsymbol{D}$, which is $(1,0)$, is assigned to share 1 and the second row of $\boldsymbol{D}$, which is $(0,1)$, is assigned to share 2. Noting that the shares are printed on

transparencies, so share 1 gets a black pixel and a transparent pixel on its transparency, and share 2 get a transparent pixel and a black pixel on its transparency. When stacking share 1 and share 2, the stacking result consists of two black pixels. Likewise, to encode the second secret pixel (i.e., a white one), we randomly choose a matrix from the set $\mathcal{C}_0$. Suppose that we get the matrix

$$\boldsymbol{D} = \begin{pmatrix} 1 & 0 \\ 1 & 0 \end{pmatrix}. \tag{5}$$

Then the first row of $\boldsymbol{D}$ is $(1,0)$ is assigned to share 1 and the second row of $\boldsymbol{D}$ is $(1,0)$ is assigned to share 2. So, share 1 gets a black pixel and a transparent pixel on its transparency, and share 2 gets the same pattern as share 1. When stacking share 1 and share 2, the stacking result consists of one black pixel and one transparent pixel. So, after stacking, it is possible to distinguish between the white pixel and the black pixel in the secret image.

Figure 2. A 3D illustration of basic $(2,2)$-threshold VC. Please note that share 1 and share 2 are printed on transparencies.

2.2. Flip VC

In 2010, Lin et al. proposed a flipping visual cryptography scheme based on non-expanded VC [25]. The general structure of a flip VC is shown in Figure 3. Two secret images, $\boldsymbol{I}_1$ and $\boldsymbol{I}_2$, are encoded to generate two shares $\boldsymbol{S}_1$ and $\boldsymbol{S}_2$. At the decoder side, if $\boldsymbol{S}_1$ and $\boldsymbol{S}_2$ are stacked, an image $\hat{\boldsymbol{I}}_1$ is generated, which reveals the content of the secret image $\boldsymbol{I}_1$. However, if we flip $\boldsymbol{S}_1$ horizontally before stacking with $\boldsymbol{S}_2$(A flipping operation on an image is a horizontal mirroring operation so that two pixels at positions (i,j) and $(i,N-1-j)$ are exchanged, where N is the number of columns of that image, and i is row index. For a more precise definition, please refer to Section 3.1), then a different image $\hat{\boldsymbol{I}}_2$ can be generated, which reveals the contents of $\boldsymbol{I}_2$. So, a flip VC can encode two different secret images into two shares simultaneously.

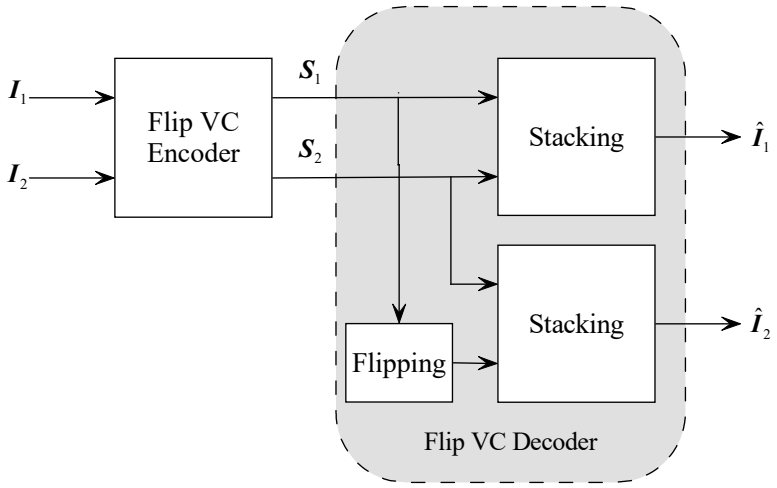


Figure 3. The block diagram of flip visual cryptography (VC).

However, more restrictions are enforced on encoder in flip VC than ordinary VC. Figure 4 shows the stacking operation by the decoder. While in conventional VC one only needs to consider one pixel a time, now we need to consider operations on four pixels $S_1(i,j), S_1(i,N-1-j), S_2(i,j), S_2(i,N-1-j)$ simultaneously, where $S_k(i,j)$ is the pixel at the (i,j) location of share image $\boldsymbol{S}_k$, and N is the number of columns. These four pixels can be grouped into a quadruple

$$\boldsymbol{s} = [S_1(i,j), S_1(i,N-1-j), S_2(i,j), S_2(i,N-1-j)].$$

To recover the secret image, the decoder must meet the following requirements:

1. Stacking $S_1(i,j)$ with $S_2(i,j)$, the secret image $I_1(i,j)$ should be revealed, i.e., $S_1(i,j) \otimes S_2(i,j) = \hat{I}_1(i,j)$, where $\otimes$ denotes the stacking operation.
2. Stacking $S_1(i,N-1-j)$ with $S_2(i,N-1-j)$, the secret image $I_1(i,N-1-j)$ should be revealed, i.e., $S_1(i,N-1-j) \otimes S_2(i,N-1-j) = \hat{I}_1(i,N-1-j)$.
3. Stacking $S_1(i,N-1-j)$ with $S_2(i,j)$, the secret image $I_2(i,j)$ should be revealed, i.e., $S_1(i,N-1-j) \otimes S_2(i,j) = \hat{I}_2(i,j)$.
4. Stacking $S_1(i,j)$ with $S_2(i,N-1-j)$, the secret image $I_2(i,N-1-j)$ should be revealed, i.e., $S_1(i,j) \otimes S_2(i,N-1-j) = \hat{I}_2(i,N-1-j)$.

The first two conditions ensure that when $\boldsymbol{S}_1$ and $\boldsymbol{S}_2$ are stacked, we can decode the secret image $\boldsymbol{I}_1$. The last two conditions ensure that when flipped $\boldsymbol{S}_1$ is stacked with $\boldsymbol{S}_2$, we can decode the secret image $\boldsymbol{I}_2$.

Lin et al. constructed 16 basic matrices for each of the quadruples

$$\boldsymbol{s} \in \{WWWW, WWWB, WWBW, \ldots, BBBB\},$$

where B denotes 'black' and W denotes 'white'. As an example, the basic matrix for quadruple 'BWBB' is [25]:

$$\boldsymbol{D}_{\text{BWBB}} = \begin{bmatrix} 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 1 & 0 \end{bmatrix} \tag{6}$$

So, if the encoder needs to encode the quadruple

$$\boldsymbol{s} = [S_1(i,j), S_1(i,N-1-j), S_2(i,j), S_2(i,N-1-j)] = BWBB, \tag{7}$$

then it randomly selects one column from the basic matrix $\boldsymbol{D}_{\mathrm{BWBB}}$ and assign it to the four pixels $S_1(i,j), S_1(i,N-1-j), S_2(i,j), S_2(i,N-1-j)$ on the two shares $\boldsymbol{S}_1$ and $\boldsymbol{S}_2$, respectively. It is not difficult to verify that, when the first row and the third row of $\boldsymbol{D}_{BWBB}$ are stacked, the ratio of black is 1. This ensures that when stacking $S_1(i,j)$ and $S_2(i,j)$, the probability of black pixel is 1. When the second row and the fourth row are stacked, the ratio of black is $5/6$. This ensures that when stacking $S_1(i,N-1-j)$ and $S_2(i,N-1-j)$, the probability of black pixel is $5/6$ and the probability of white pixel is $1/6$. So a relative contrast of $1/6$ can be ensured on stacking results. For a complete list of the 16 basic matrices, please refer to Table 1 in [25]. Lin also proved that using these basic matrices, a relative contrast of $1/6$ can be ensured on stacking results. Furthermore, his construction is also proved to be secure.

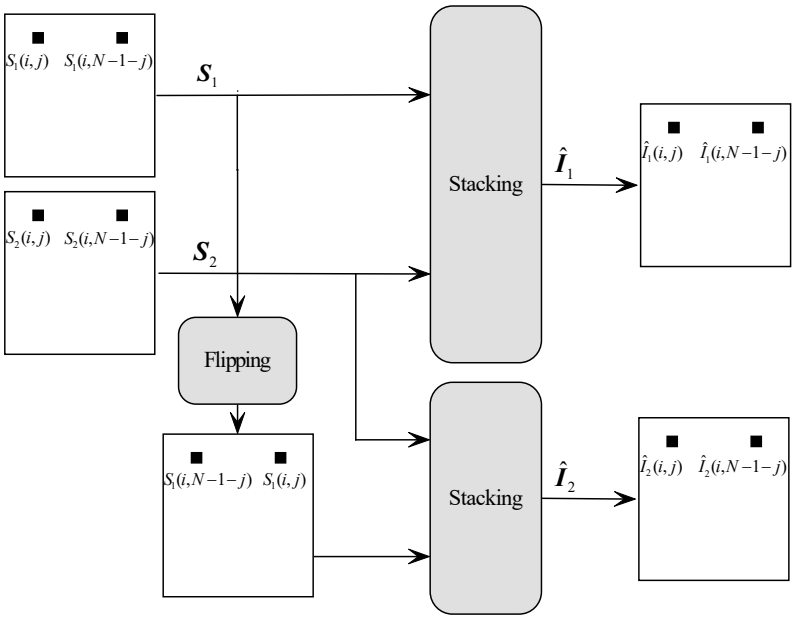


Figure 4. Restrictions imposed on shares by flip VC decoder.

3. Proposed Scheme

Before elaborating on the proposed algorithm, let us fix the notations for matrix operation and share stacking.

3.1. Definition of Operation

Definition 1 (matrix concatenation)**.** *Let $\boldsymbol{y}(i,j)$ be one of the matrices in the share matrix, where i,j are the position indices of the block matrix in the share matrices. Define*

$$\boldsymbol{y} = \boxplus_{i=1}^{g} \boxplus_{j=1}^{h} \boldsymbol{y}(i,j) \tag{8}$$

as the concatenation of matrices. For example, if

$$\boldsymbol{y}(1,1) = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \ \boldsymbol{y}(1,2) = \begin{pmatrix} 1 & 0 \\ 1 & 0 \end{pmatrix} \tag{9}$$

then we have

$$\boldsymbol{y} = \boldsymbol{y}(1,1) \boxplus \boldsymbol{y}(1,2) = \begin{pmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 \end{pmatrix}. \tag{10}$$

Definition 2 (stacking)**.** *Operation for image stacking* $\otimes$ *is the logical OR operate, where* $1 \otimes 1 = 1, 0 \otimes 1 = 1, 1 \otimes 0 = 1, 0 \otimes 0 = 0$*. For two matrices* $\boldsymbol{A}$ *and* $\boldsymbol{B}$*,* $\boldsymbol{A} \otimes \boldsymbol{B}$ *results in pixel-wise OR between two pixels at the same position on* $\boldsymbol{A}$ *and* $\boldsymbol{B}$*.*

Definition 3 (matrix flipping)**.** *Matrix flipping operation* $B = \mathcal{F}(\boldsymbol{A})$ *flips the matrix* $\boldsymbol{A}$ *with respect to its central column. Let* $\boldsymbol{A} \in \mathbb{R}^{m \times n}$*, where* $\mathbb{R}$ *is the set of real numbers, and m and n are even integers. Then we have*

$$B(i, j) = A(i, n - 1 - j), \tag{11}$$

where $0 \leq i \leq m - 1, 0 \leq j \leq n - 1$*. For example, given*

$$\boldsymbol{y} = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix},$$

and then we have

$$\mathcal{F}(\boldsymbol{y}) = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}.$$

The flipping operation is part of the VC decoder presented in the next subsection.

Definition 4 (Left and right region of a matrix)**.** *Given a matrix* $\boldsymbol{A}$*, we define the operation* $\mathcal{L}(\boldsymbol{A})$ *as returning the left half region of the matrix* $\boldsymbol{A}$*:*

$$\mathcal{L}(\boldsymbol{A}) = \{(i, j) \mid 0 \leq i \leq M - 1, 0 \leq j \leq N/2 - 1\}. \tag{12}$$

Similarly, we define the operation $\mathcal{R}(\boldsymbol{A})$ *as returning the right half region of the matrix* $\boldsymbol{A}$*:*

$$\mathcal{R}(\boldsymbol{A}) = \{(i, j) \mid 0 \leq i \leq M - 1, N/2 \leq j \leq N - 1\}. \tag{13}$$

Obviously, when concatenating the two halves, we should have $\boldsymbol{A}(\mathcal{L}(\boldsymbol{A})) \boxplus \boldsymbol{A}(\mathcal{R}(\boldsymbol{A})) = \boldsymbol{A}$*.*

3.2. Overview

In this section, we describe the overall structure of the proposed algorithm. The block diagram of this system is shown in the Figure 5. During encoding stage, two secret images $\boldsymbol{I}_1$ and $\boldsymbol{I}_2$ are encrypted into two shares $\boldsymbol{S}_1$ and $\boldsymbol{S}_2$ that are perceptually similar to the two cover images $\boldsymbol{C}_1$ and $\boldsymbol{C}_2$.

These shares are distributed to two different participants. During decoding stage, after gathering two shares, the decoder is able to recover the content of the secret images by stacking operation. More specific, the content of secret image $\hat{\boldsymbol{I}}_1$ can be recovered by stacking $\boldsymbol{S}_1$ and $\boldsymbol{S}_2$:

$$\boldsymbol{S}_1 \otimes \boldsymbol{S}_2 = \hat{\boldsymbol{I}}_1, \tag{14}$$

and the content of the secret image $\hat{\boldsymbol{I}}_2$ can be recovered by stakcing $\boldsymbol{S}_1$ with the flipped $\boldsymbol{S}_2$

$$\boldsymbol{S}_1 \otimes \mathcal{F}(\boldsymbol{S}_2) = \hat{\boldsymbol{I}}_2, \tag{15}$$

where $\mathcal{F}(\boldsymbol{S}_2)$ is the flipping operation as defined in previous subsection.

We can recover the secret image $\hat{\boldsymbol{I}}_1$ that have the same information content as $\boldsymbol{I}_1$ when stacking the front side of share $\boldsymbol{S}_1$ with the front side of share $\boldsymbol{S}_2$. However, when we stack the front side of share $\mathbf{S}_1$ and the reverse side of share $\boldsymbol{S}_2$, we can recover the secret image $\hat{\boldsymbol{I}}_2$.

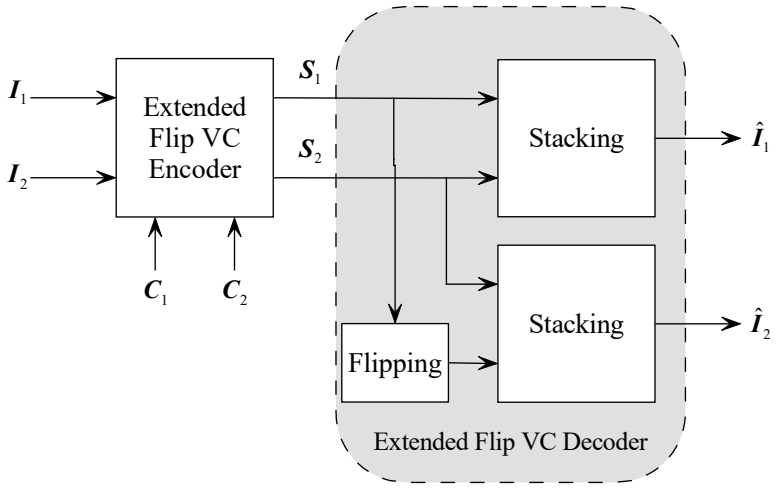


Figure 5. Overall structure of the system. $\boldsymbol{C}_1$ and $\boldsymbol{C}_2$ are cover images, $\boldsymbol{I}_1$ and $\boldsymbol{I}_2$ are secret images, $\boldsymbol{S}_1$ and $\boldsymbol{S}_2$ are share images, $\hat{\boldsymbol{I}}_1$ and $\hat{\boldsymbol{I}}_2$ are recovered images.

3.3. Determining Pixel Position

Let us assume in this section that the cover images are gray-scale images and the secret images are binary images. The grayscale cover image $\boldsymbol{C}_k, k \in \{1,2\}$ is of M rows and N columns. We divide the share images into blocks of size $Q \times Q$, and encode the 1-bit of (i.e., 1 pixel) secret image into $Q \times Q$ block in the share images. So the size of the secret images $\boldsymbol{I}_k, k \in \{1,2\}$ has $g = \lfloor \frac{M}{Q} \rfloor$ rows and $h = \lfloor \frac{N}{Q} \rfloor$ columns, where $\lfloor x \rfloor$ returns the nearest integer towards the $-\infty$.

When stacking two shares in order to recover the secret image, two types of interference may appear:

- Interference from the cover images.
- Interference from the other secret image.

In order to recover secret images without interference, we must superimpose pixels that do not contain secret information in black. Let the matrix $\boldsymbol{M}_p, p \in \{1,2\}$ contain all the secret image pixels and the matrix $\boldsymbol{B}_q, q \in \{1,2\}$ contain all the auxiliary black pixels, where p is the index of the matrix of the secret image informations and q is the index of the share matrix. When recovering a secret image, in order to avoid interference of another secret image pixel, another secret pixel must be black after stacking. We define these pixels as auxiliary secret pixels $\boldsymbol{F}$.

In our algorithm, the positions for encoding the secret pixels are generated randomly. This random approach is advantageous than deterministic approaches in terms of security and ensuring the exact number of secret image pixels (SIP). The position determination algorithm is summarized in Algorithm 1.

We assume that each share image is segmented into $Q \times Q$ blocks. Within each block, there are 2α black pixels, which means that α pixels are used for encoding the secret pixel from $\boldsymbol{I}_1$ and α pixels are used for encoding the secret pixel from $\boldsymbol{I}_2$.

When stacking two shares, we must use auxiliary black pixels (ABP) to remove the interference from the non-SIPs on stacking result. To ensure that we have enough ABPs, then number of ABPs should be at least half of the number of non-SIPS, i.e.,

$$\beta = \lceil \frac{Q^2 - 2\alpha}{2} \rceil. \tag{16}$$

Algorithm 1 Determine pixel position

Input:

Q: Size of block.

$M \times N$: Size of the share image.

α: The number of pixels in each secret image in a block.

β: The number of black auxiliary pixels in a block.

Output:

$\boldsymbol{M}_1$ and $\boldsymbol{M}_2$: Position matrix of pixel locations of secret image $\boldsymbol{I}_1$ and secret image $\boldsymbol{I}_2$ pixels in the share image.

$\boldsymbol{B}_1$ and $\boldsymbol{B}_2$: Position matrix of black auxiliary pixel position in share image $\boldsymbol{S}_1$ and $\boldsymbol{S}_2$.

for $i \leftarrow 1$ to $\lfloor \frac{M}{Q} \rfloor$ **do**
  for $j \leftarrow 1$ to $\lfloor \frac{N}{Q} \rfloor$ **do**
    Generate Q^2 non-repeating random numbers from 1 to Q^2 to form a sequence $\boldsymbol{p}$, and map the sequence to a $Q \times Q$ pixel block.
    Select the first 2α random numbers and return their corresponding positions in the pixel block $\boldsymbol{x}(k)$, $\boldsymbol{y}(k)$, Where $\boldsymbol{x}(k)$ and $\boldsymbol{y}(k)$ represent the row and column positions of the kth random number in the pixel block, respectively.
    for $m \leftarrow 1$ to α **do**
      $\boldsymbol{M}_1((i-1) \times Q + \boldsymbol{x}(k), (j-1) \times Q + \boldsymbol{y}(k)) \leftarrow 1$
    end for
    for $m \leftarrow \alpha$ to 2α **do**
      $\boldsymbol{M}_2((i-1) \times Q + \boldsymbol{x}(k), (j-1) \times Q + \boldsymbol{y}(k)) \leftarrow 1$
    end for
    Select the random number after the 2αth and return its corresponding position $\boldsymbol{x}(k)$, $\boldsymbol{y}(k)$ in the pixel block.
    for $m \leftarrow 2\alpha + 1$ to $2\alpha + \beta$ **do**
      $\boldsymbol{B}_1((i-1) \times Q + \boldsymbol{x}(k), (j-1) \times Q + \boldsymbol{y}(k)) \leftarrow 1$
    end for
    for $m \leftarrow 2\alpha + \beta + 1$ to end **do**
      $\boldsymbol{B}_2((i-1) \times Q + \boldsymbol{x}(k), (j-1) \times Q + \boldsymbol{y}(k)) \leftarrow 1$
    end for
  end for
end for
Return the position matrices $\boldsymbol{M}_1, \boldsymbol{M}_2, \boldsymbol{B}_1, \boldsymbol{B}_2$

First, to determine the location for SIPs and non-SIPs, we generate a vector $\boldsymbol{p}$ which contains a permutation of all integers between 1 and Q^2. Then we use the first α numbers as positions of the SIPs for $\boldsymbol{I}_1$, and use the numbers $\boldsymbol{p}(\alpha) \cdots \boldsymbol{p}(2\alpha)$ as positions of the SIPs for $\boldsymbol{I}_2$. The positions $\boldsymbol{p}(2\alpha + 1) \cdots \boldsymbol{p}(2\alpha + \beta)$ are for ABPs in share $\boldsymbol{S}_1$, and the positions $\boldsymbol{p}(2\alpha + \beta + 1) \cdots \boldsymbol{p}(Q^2 - 1)$ are for ABPs on share $\boldsymbol{S}_2$. Next, we map these positions into positions in a $Q \times Q$ block, and fill the corresponding positions of $\boldsymbol{M}_1, \boldsymbol{M}_2, \boldsymbol{B}_1, \boldsymbol{B}_2$ with 1 and other positions with 0.

For example, let us take $Q = 4, \alpha = 2$, then we have $\beta = 6$. Let us assume that the random permutation is

$$\boldsymbol{p} = (1, 6, 11, 16, 2, 7, 3, 14, 13, 9, 4, 12, 5, 8, 15, 10). \tag{17}$$

Then, the positions $(1, 6)$ are used for embedding pixel from $\boldsymbol{I}_1$, and the positions $(11, 16)$ are used to embed pixel from $\boldsymbol{I}_2$. The positions for ABPs on share $\boldsymbol{S}_1$ and $\boldsymbol{S}_2$ are $(2, 7, 3, 14, 13, 9)$ and $(4, 12, 5, 8, 15, 10)$, respectively. Suppose that the current image block index is (k, l), then we get the following position matrices:

$$\begin{aligned} \boldsymbol{M}_1(k,l) &= \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}, \quad \boldsymbol{M}_2(k,l) = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \\ \boldsymbol{B}_1(k,l) &= \begin{pmatrix} 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}, \quad \boldsymbol{B}_2(k,l) = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix} \end{aligned} \tag{18}$$

These positions are also illustrated in Figure 6.

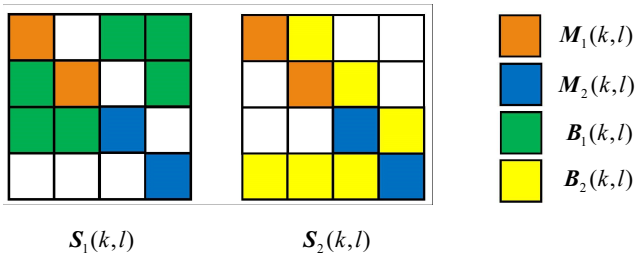


Figure 6. Determining the locations of secret pixels and non-secret pixels($Q = 4, \alpha = 2, \beta = 6$).

3.4. Embedding the Secret Image into Shares

From the last subsection, we get a collection of secret pixel locations and non-secret pixel locations. In this section, we describe the process of embedding the secret information at the secret pixel locations. Since every $Q \times Q$ block in the share contains two pieces of secret information and corresponds to another share in different ways, the secret pixel position needs special design when embedding the secret pixel.

In order to eliminate interference during stacking, we have to adjust the position matrices obtained from previous section before embedding secret pixels. The ABP matrix $\boldsymbol{F}$ must meet the following two requirements

1. When stacking $\boldsymbol{S}_1$ and $\boldsymbol{S}_2$, the ABP matrix $\boldsymbol{F}$ should cover the SIPs for the secret image $\boldsymbol{I}_2$;
2. When stacking $\boldsymbol{S}_1$ and $\mathcal{F}(\boldsymbol{S}_2)$, the ABP matrix $\boldsymbol{F}$ should cover the SIPs for the secret image $\boldsymbol{I}_1$.

So, we should adjust the position matrices as follows.

$$\boldsymbol{F} = \mathcal{F}(\boldsymbol{M}_1), \tag{19}$$
$$\boldsymbol{M}_1 = \mathcal{L}(\boldsymbol{M}_1) \boxplus \mathcal{F}(\mathcal{L}(\boldsymbol{M}_2)), \tag{20}$$
$$\boldsymbol{M}_2 = \mathcal{F}(\boldsymbol{M}_1), \tag{21}$$
$$\boldsymbol{B}_1 = \mathcal{L}(\boldsymbol{B}_1) \boxplus \mathcal{F}(\mathcal{L}(\boldsymbol{B}_2)), \tag{22}$$
$$\boldsymbol{B}_2 = \mathcal{L}(\boldsymbol{B}_2) \boxplus \mathcal{F}(\mathcal{L}(\boldsymbol{B}_1)). \tag{23}$$

These positions on shares S_1 and S_2 are illustrated in Figure 7.

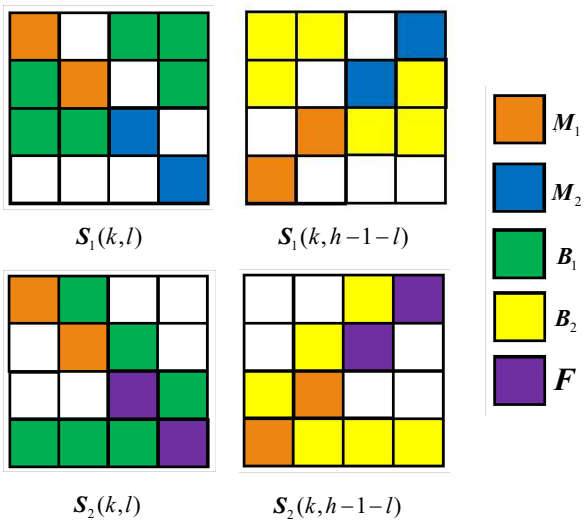


Figure 7. Locations on shares after adjustment.

The basic matrices for bit 1 and bit 0 are denoted as D_0 and D_1, respectively. For example, for a (2, 2) scheme, when $\alpha = 2$, we let

$$D_0 = \begin{pmatrix} 1 & 0 \\ 1 & 0 \end{pmatrix}, D_1 = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}. \tag{24}$$

That is, when the secret pixel is bit 0, then D_0 is selected and it is transformed into $\Pi(D_0)$, where Π denotes column permutation. The first line of $\Pi(D_0)$ is filled into the corresponding secret pixel position in share 1, and the second row is filled in the corresponding secret pixel in share S_2.

Now that we know how to embed a secret bit, then we can embed the whole secret image into the shares. First, we embed the bits from I_1. The other block index is (*k*,*l*), which is also the pixel index of the secret pixel. If the pixel $M_1(k,l)$ in I_1 is 0, we need to fill the lines of $\Pi(D_0)$ into the corresponding M_1 positions in the two shares, otherwise we fill in $\Pi(D_1)$. Next, the bits from I_2 is embedded based on the information of I_1. If the bit $M_2(k,l)$ of I_2 is 0, we fill in the complement of the $M_1(k,l)$ position information in S_2 into the $M_2(k, \lfloor \frac{N}{Q} \rfloor - l)$ position in S_1. Instead, the $M_1(k,l)$ location information is copied. Finally, fill the auxiliary secret pixel F and the auxiliary black pixel B_q position with the bit 1.

3.5. Embed Cover Image In Share

The above section describes the process of embedding two secret images into two shares. The share obtained through this process is meaningless. In this section, we use constraint error diffusion to embed the cover image into the shares [26], making the shares meaningful. The pixels in the share image are sequentially scanned using the raster scanning order. The cover image is quantized and embedded into the share image. The quantization process can be expressed as

$$\begin{aligned} S_t(k,l) = M_p(k,l) \cdot B_q(k,l) \cdot F(k,l) \cdot S_t(k,l) \\ +(1 - M_p(k,l)) \cdot (1 - B_q(k,l)) \cdot (1 - F(k,l)) \cdot Q(C_q(k,l)), \end{aligned} \tag{25}$$

where $M_p, p \in \{1,2\}$, $B_q, q \in \{1,2\}$, and F are the position matrices obtained from previous sections. The first term in the summation says that for those positions carrying the SIPs and ABPs, we leave them unchanged during cover image embedding process. The second

terms says that for those positions allocated for carrying cover image, we use a simple binary quantizer:

$$Q(\boldsymbol{C}_q(k,l)) = \begin{cases} 1, & \text{if } \boldsymbol{C}_q(k,l) \geq 127 \\ 0, & \text{if } \boldsymbol{C}_q(k,l) < 127 \end{cases} \tag{26}$$

To achieve better visual quality for the share images, the error between the share images and the cover images is spread to the non-secret information positions and the non-auxiliary positions by the error diffusion. Error diffusion diffuses the error produced by quantization at current location to its neighbors, so that when quantizing these neighboring pixels, these errors can be compensated. A diffusion kernel is like a filter kernel that determines how the errors are allocated to its neighbors [24].

Let $h(k,l)$ be the diffusion kernel located in the pixel (k, l), then in the constrained diffusion, the modified diffusion coefficient is $h(k,l) = h(k,l) \cdot (1 - \boldsymbol{M}_p) \cdot (1 - \boldsymbol{B}_q) \cdot (1 - \boldsymbol{F})$. That is, the diffusion coefficient is set to 0 at the $\boldsymbol{M}_p$ and $\boldsymbol{B}_q$ and $\boldsymbol{F}$ positions.

4. Flip Visual Cryptography for Color Images

Color visual cryptography has better visual quality than its gray-scale counterpart since it can render more details with colors. In this section, we design a color VC in RGB color space. The secret image is divided into three channels of R, G, and B, and the secret image is embedded into the share image according to the principle of color stacking. The color overlay model is shown in Figure 8.

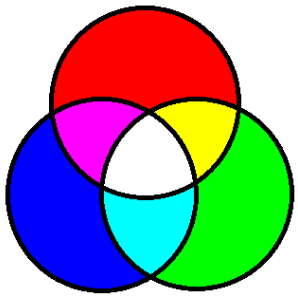


Figure 8. Color overlay model for RGB color space.

Our color VC is based on the human perception of color. When different pixels having different colors are clustered then our HVS (Human Visual System) perceives and average color. For example, if four pixels having color R, G, B, and W (R: red, G: green, B: blue, and W: white) are arranged as a 2×2 block, then we perceive white. Similarly, the effect of other colors can be achieved according to the proportion of colors in a small block. For example, if we have two B, one R and one G, then the ratio of the R, G, and B colors that are shared by the two shares is $\left(\frac{1}{4}, \frac{1}{4}, \frac{1}{2}\right)$, and the perceived color is blue. We can get different proportions of different colors after superimposing the sharing of different colors, as shown in Figure 9.

The steps to embed a secret image in a share during the color visual cryptography process are as described as follows:

1. Step 1: Color decomposition and halftoning. The RGB color image channels of $\boldsymbol{I}_1$ and $\boldsymbol{I}_2$ are separately halftoned.
2. Step 2: Embed R, G, B, W at the secret information 1 position in the $\boldsymbol{S}_1(k, l)$, where (k, l) denotes the position of the pixel block, where $k \leq g, l \leq h$.
3. Step 3: Embed the corresponding color in the secret information $\boldsymbol{I}_1$ position of the $\boldsymbol{S}_2(i, j)$ pixel block according to the color of $\boldsymbol{I}_1(i, j)$ and the color of $\boldsymbol{S}_1(i, j)$ position corresponding to secret information $\boldsymbol{I}_1$.

4. Step 4: Embed in the secret information $\boldsymbol{I}_2$ position of the pixel block at $\boldsymbol{S}_1(i, N+1-j)$ position according to the color of $\boldsymbol{I}_2(i, N+1-j)$ and the color of the secret information 1 of $\boldsymbol{S}_2(i, j)$ pixel block.
5. Step 5: Embed R, G, B, and W at the secret information $\boldsymbol{I}_1$ position in $\boldsymbol{S}_1(i, N+1-j)$ position pixel block.
6. Step 6: Fill in secret information $\boldsymbol{I}_1$ position of $\boldsymbol{S}_2$ $(i, N+1-j)$ pixel block according to the color of $\boldsymbol{I}_1(i, N+1-j)$ and the color of secret information 1 in $\boldsymbol{S}_1(i, N+1-j)$ pixel block.
7. Step 7: According to the color of $\boldsymbol{I}_2$ (i, j) and the color of secret information $\boldsymbol{I}_1$ in $\boldsymbol{S}_2$ $(i, N+1-j)$ pixel block, fill in the corresponding color of the secret information $\boldsymbol{I}_2$ position in $\boldsymbol{S}_1$ (i, j) pixel block.

Secret pixel color	S_1 Pixel Block Color	S_2 Pixel Block Color	Recover secret pixel block
R	$\begin{pmatrix} R & G \\ B & W \end{pmatrix}$	$\begin{pmatrix} W & G \\ B & R \end{pmatrix}$	$\begin{pmatrix} R & G \\ B & R \end{pmatrix}$
G	$\begin{pmatrix} R & G \\ B & W \end{pmatrix}$	$\begin{pmatrix} R & W \\ B & G \end{pmatrix}$	$\begin{pmatrix} R & G \\ B & G \end{pmatrix}$
B	$\begin{pmatrix} R & G \\ B & W \end{pmatrix}$	$\begin{pmatrix} R & G \\ W & B \end{pmatrix}$	$\begin{pmatrix} R & G \\ B & B \end{pmatrix}$
C	$\begin{pmatrix} R & G \\ B & W \end{pmatrix}$	$\begin{pmatrix} R & B \\ G & W \end{pmatrix}$	$\begin{pmatrix} R & C \\ C & W \end{pmatrix}$
M	$\begin{pmatrix} R & G \\ B & W \end{pmatrix}$	$\begin{pmatrix} B & G \\ R & W \end{pmatrix}$	$\begin{pmatrix} M & G \\ M & W \end{pmatrix}$
Y	$\begin{pmatrix} R & G \\ B & W \end{pmatrix}$	$\begin{pmatrix} G & R \\ B & W \end{pmatrix}$	$\begin{pmatrix} Y & Y \\ B & W \end{pmatrix}$
K	$\begin{pmatrix} R & G \\ B & W \end{pmatrix}$	$\begin{pmatrix} R & G \\ B & W \end{pmatrix}$	$\begin{pmatrix} R & G \\ B & W \end{pmatrix}$
W	$\begin{pmatrix} R & G \\ B & W \end{pmatrix}$	$\begin{pmatrix} W & B \\ G & R \end{pmatrix}$	$\begin{pmatrix} R & C \\ C & R \end{pmatrix}$

Figure 9. Color sub-channel visual cryptography scheme.

5. Experiments

5.1. Experiment for Gray-Scale Image

In this section, we show the experimental results for gray-scale cover images. The parameters used for the experiment are $M = N = 768, Q = 4, \alpha = 2$ and $\beta = 6$. In order to measure the visual quality of the share images, we use the MSSIM (Mean Structural Similarity Measure) and PSNR (Peak Signal-to-Noise Ratio) metrics. The diffusion mask is the Floyd—Steinberg mask in experiments. The two cover images and two secret images are shown in Figure 10.

The share images obtained from the proposed algorithm are shown in Figure 11. Before calculating image quality, Gaussian filtering is performed on the halftone image of the cover images and the share images. This Gaussian filtering simulates the low-pass property of the HVS. The PSNRs for the two share images are 16.3478 and 13.8942, respectively. The MSSIM measures are 0.6584 and 0.6196, respectively.

(**a**) (**b**) (**c**) (**d**)

Figure 10. The images used in the experiment. (**a**–**d**) are cover image C_1, cover image C_2, secret image I_1, and secret image I_2, respectively.

(**a**) (**b**)

Figure 11. The generated meaningful shares. (**a**,**b**) are share image S_1, share image S_2, respectively. The corresponding PSNR values are 16.3478 and 13.8942, respectively.

The superimposed results of the two share images are shown in Figure 12. Among them, Figure 12a is stacking result of Figure 11a,b. Figure 12b is the stacking result of the front side of Figure 11a and the reverse side of Figure 11b.

(**a**) (**b**)

Figure 12. The superimposed results of the two share images. (**a**) The recovered secret image 1. (**b**) The recovered secret image 2.

In order to test if the proposed algorithm is applicable to various types of images, we performed a batch test on ten standard testing images. Figure 13 shows the gray-scale cover images and Figure 14 shows the secret images. In our experiment, different images are selected as the cover images, and the corresponding binary image with the same size as cover images are used as the secret image. The binary secret images are randomly chosen from a set of 18 binary images. The grayscale cover images are randomly chosen from a set of 18 grayscale images. If the size of the secret image is too large or too small, it is resized so that it is suitable for the cover image. To resize the binary image, we found that

the nearest interpolation method gives a reasonably good result. Then, we use the simple binary thresholding to convert the gray-scale image to a binary image. PSNR and MSSIM metrics are used to measure the visual quality of the share images.

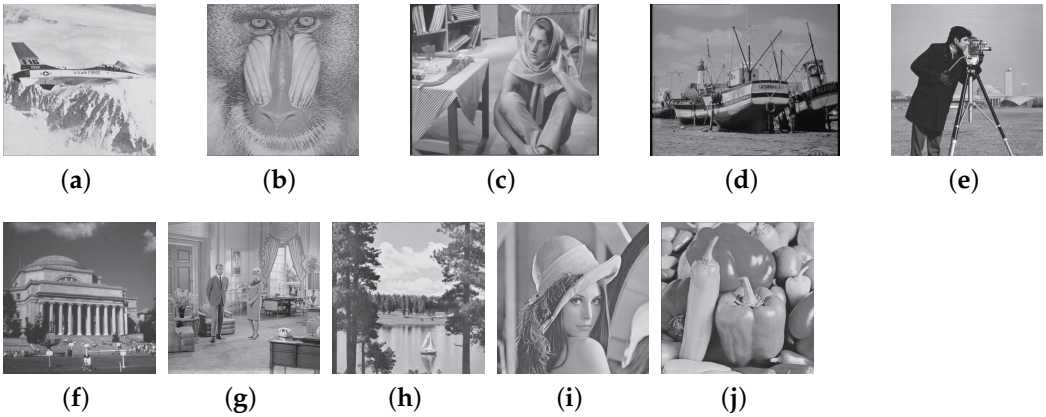


Figure 13. A collection of cover images in batch test. (**a**–**j**) are samples of the cover images.

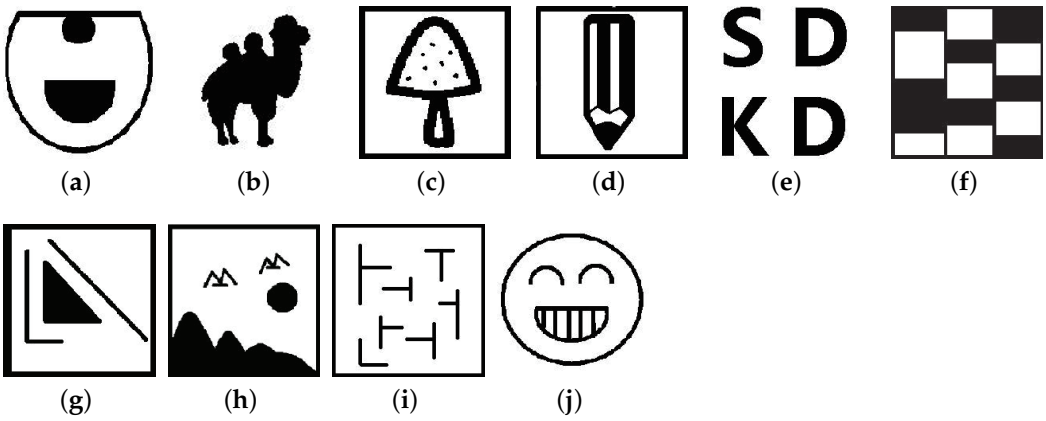


Figure 14. A collection of secret images in batch test. (**a**–**j**) are samples of the secret images.

5.1.1. Feasibility Experiment for Gray-Scale Cover Images

For the batch test, the variance of the Gaussian filter is set to 4 and the kernel size is 11×11. Table 1 shows the values of the PSNR and MSSIM values for some images in the batch test.

Table 1. The values of PSNR and MSSIM for some images in the batch test. The images kodim01 to kodim18 are cover images, and the binary images binary1 to binary18 are secret images.

C_1	C_2	S_1	S_2	PSNR1	PSNR2	MSSIM1	MSSIM2
kodim01	kodim02	binary1	binary2	16.0380	19.8768	0.6460	0.6934
kodim03	kodim04	binary3	binary4	16.2302	16.3874	0.6672	0.6548
kodim05	kodim06	binary5	binary6	17.2594	10.9287	0.6839	0.5822
kodim07	kodim08	binary7	binary8	16.1168	11.9655	0.6574	0.5601
kodim09	kodim10	binary9	binary10	13.4682	13.8673	0.6086	0.6086
kodim11	kodim12	binary11	binary12	17.8289	9.9372	0.6735	0.5284
kodim13	kodim14	binary13	binary14	14.2886	15.8585	0.6250	0.6426
kodim15	kodim16	binary15	binary16	10.7148	15.1861	0.6221	0.6402
kodim17	kodim18	binary17	binary18	17.8349	18.5478	0.6553	0.6694

It can be seen from Table 1 that, the values of PSNR and MSSIM in the image experiment do not fluctuate much, indicating that our algorithm performs well and consistent for

different cover images and secret images. The contrast of the recovered secret image is the same as in [26]. The cover image, which is also halftoned before encryption, is Gaussian filtered and compared with the share image. In this paper, the compared cover image information in each pixel block of the share image accounts for $(Q^2 - 2\beta)/Q^2$, while in [26], the compared cover image information in each pixel block of the share image accounts for $(Q^2 - \beta)/Q^2$. Therefore, the contrast of our flip VC is only slightly inferior to [26], but visually quite similar.

The time complexity of our algorithm is tested in terms of machine time. The machine time of each experiment is affected by the location of the image pixels and the random distribution of the secret information. That is to say, because the pixel positions of each image are different and the pixel positions are randomly distributed when the secret information is embedded, the machine time of each image experiment is also different. The machine time of each experiment in the batch experiment is shown in Figure 15.

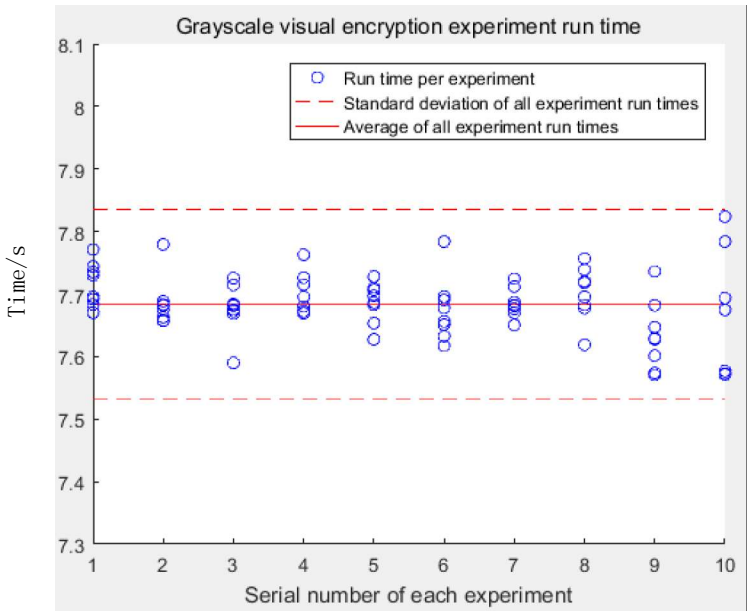


Figure 15. Machine time in batch test.

According to the chart, it can be seen that the machine time of the algorithm has nothing to do with the content of the image, it is only related to the size of the image, all floating around the average. At the same time, it can be seen that the short machine time of the algorithm is generally about 7.7 s, indicating that the algorithm is simple to run, high in efficiency, and easy to apply.

5.1.2. Security Analysis

This section analyzes the security of our algorithm. We show that it is not possible to infer the secret image from just one share. Each 4×4 block in the share image contains two secret image bits and two cover image bits, each of which has the same proportion of secret image information. At the same time, the position of each pixel information is random. In Figure 16, we list the position distribution of the corresponding pixel blocks in the share image $\mathbf{S}_1$ between the secret image $\mathbf{I}_1(i,j)$ and the secret image $\mathbf{I}_2(i,j)$. The five cases are listed among all the possibilities. Among them, $\mathbf{I}_1(i,j)$ and $\mathbf{I}_2(i,j)$ are secret images, (i,j) is a pixel index of the secret image, $\mathbf{S}_1(k,l)$ is a share image, and (k,l) is a block index corresponding to the secret image (i,j). Each position situation has the same probability.

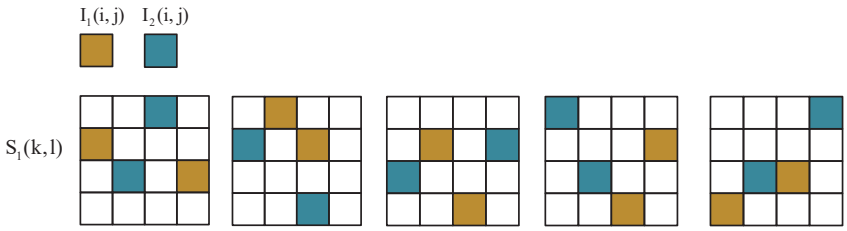


Figure 16. The distribution of secret image bits in the share image pixel block (five in all cases).

The secret information in the corresponding pixel block in the share image is obtained by the random column transformation of the basic matrix. The probability of each position situation in the picture is the same. so the secret information in the pixel block cannot be determined. Therefore, even if someone obtains one of the share images, the secret image information cannot be obtained from it. Therefore, the system build using our algorithm is secure.

In our algorithm, the share images use a random allocation method to embed the secret image and the cover image. So, the attacker cannot copy the secret information and the black auxiliary information location without knowing the allocation location. An attacker can only change any pixel information in a pixel block. However, under this attack, the secret image can still be recovered. Therefore, the proposed algorithm is also robust to attacks.

5.2. Experiments for Color Images

In this section, we show the experimental results for color cover images. The parameters used for the experiment are $M = N = 504, Q = 4, \alpha = 4$ and $\beta = 4$. In order to measure the quality of the share images, we apply the MSSIM and PSNR parameters. The diffusion kernel is the Floyd—Steinberg mask in experiments. The two cover images and two secret images are shown in Figure 17.

(**a**) (**b**) (**c**) (**d**)

Figure 17. The images used in the experiment. (**a**–**d**) are cover image C_1, cover image C_2, secret image I_1, and secret image I_2, respectively.

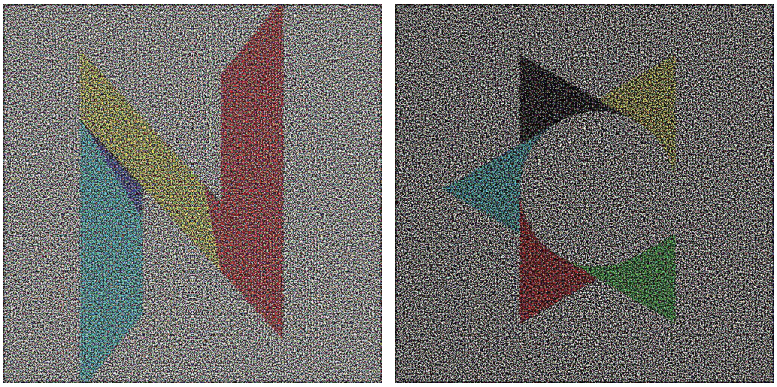
The color flip visual cryptography experiment is similar to the gray-scale flip visual cryptography. When the two share images are superimposed on the front side, the secret image I_1 is recovered. After the share image S_2 is flipped and the share image S_1 is stacked, we obtain the secret image I_2. The experimental result is shown in Figure 18.

The secret image recovery result is shown in Figure 19, where Figure 19a is the recovery secret image $\hat{I}_1$, and Figure 19b is the recovery secret image $\hat{I}_2$.

5.2.1. Testing Contrast

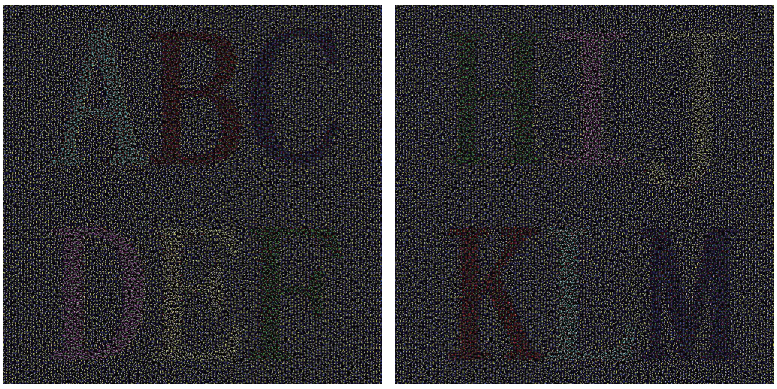
According to the algorithm principle of the previous chapter, when the two share images are superimposed to generate a secret image, the recovered black color is biased, not pure black, but a black-like color displayed by mixing cyan and red through the human eye system. At the same time, because the secret pixel information in each pixel of the share image is relatively small, the visual quality of the recovered secret image is dark. The size of each pixel block in the paper is 4×4. According to the principle of visual cryptography and overlay, it can be concluded that only 4 pixels in the share image pixel block are one

of the secret image information. The color obtained by superimposing the secret image information is a mixture of colors, so only two pixels in each share image pixel block are secret image information colors. That is, the contrast of the share image with respect to the cover image is 1/8.



(**a**) (**b**)

Figure 18. The generated meaningful shares. (**a**,**b**) are share image1 and share image2 S_2, respectively. PSNR values are 3.5139 and 2.5697, respectively.



(**a**) (**b**)

Figure 19. The superimposed results of the two share images. (**a**) The recovered secret image 1. (**b**) The recovered secret image 2.

Because the pixel block in each superimposed recovered secret image includes information auxiliary pixels and black auxiliary pixels, and each auxiliary pixel is a black pixel, the entire recovered secret image is displayed black, and the black and white information of the secret image is not obvious. Because 3/4 of the pixel blocks in each share are superimposed with black pixels and white pixels are mixed with four colors of R, G, B, and W, the superimposed white pixels are relatively dark and cannot be used. This analysis shows that black and white pixels in a secret image can be distinguished in a recovered image.

According to the above analysis, each color in the recovered secret image has the same proportion in the block, so the brightness is the same. However, the restored images of different colors have different visual quality, objectively because the color contrast of each color and the background black is different, so each recovered secret image has a different visual quality.

Because the recovered secret images have the same brightness, we cannot objectively evaluate which color has the best visual quality, so we performed a series of subjective evaluations. First, we use the image in Kodak as a cover image and combine different letters or numbers to form a secret image for encryption. Each secret image has six different colors: red, green, blue, cyan, magenta, and yellow. In order to print the image with better

visual quality, we set the size of each secret image to 300 × 300, then cover up the image size to 1200 × 1200, and print the image with 600 dpi (dots per inch) to get a larger image, which is easier for visual inspection. Secondly, the different colors of each secret image are encrypted and superimposed to recover the secret image. The recovered secret image has a dpi of 600 and a size of two inches (5.08 cm). Finally, the different color overlay results of each secret image are printed on a piece of paper. In order to avoid the human eye's adaptation to their color differences, each experiment is repeated for a different participant. The participant is asked to be exposed to the test image for 5 s and is asked to scale the quality. These subjective rankings are then counted, and the final ranking obtained according to the principle of majority voting is used as the final subjective evaluation data. After the above experimental process, we finally get the subjective judgment results of recovering the secret image from good to bad, which are blue, red, cyan, yellow, magenta, and green, respectively. Figure 20 shows the results of a set of experiments in a batch experiment. We can sort the following experiments according to the above process.

5.3. Comparison with Other Algorithm

In our design, we apply the following restrictions to our algorithms:

1. Meaningful shares;
2. Simple decoding without calculation;
3. No interference from cover images on stacking result;
4. Applicable to color cover images.

Our experimental test verified that these requirements are met simultaneously by our design. For a fair comparison with existing algorithms, we focus on the type where no computation is needed in the decoding stage [25–29]. Among these algorithms, Lin's proposal is closely related to our proposal since both of these two belongs to flip VC [25]. Compared with Lin's algorithm, our algorithm applies to color cover images and produces meaningful shares, where Lin's algorithm is designed for binary secret image and it produces meaningless shares. In terms of computational complexity, our algorithm is slightly higher than Lin's algorithm because local error propagation is a neighborhood operation, while encoding in Lin's algorithm is point-wise. However, both algorithms have a complexity which scales linearly with the number of pixels in secret image.

Similar to our proposal, Fang's algorithm also hides a second secret image in shares which can be decoded by stacking one share with a shifted version of the other share [27]. However, Fang's algorithm is not directly applicable to color cover images and it produces meaningless shares. Furthermore, Fang's algorithm is not size-invariant so that each share image is four times larger than the secret images. In contrast, our algorithm is size-invariant. Our algorithm has a higher complexity since both the secret images and the cover images needs to be consider during encoding stage. This is the price we have to pay for added function.

Yan's algorithm hides a second secret image in meaningful shares [26]. But unfortunately, it is not directly applicable to color secret images. Yan's algorithm have a similarity complexity as our algorithm since both of them are based on constrained error diffusion. Wang's algorithm produces meaningful shares but is also not directly applicable to color secret images [28]. Furthermore, Wang's algorithm hides only one secret image, while our algorithm hides two. In terms of complexity, our algorithm has a slightly higher complexity than Wang's due to the added functions.

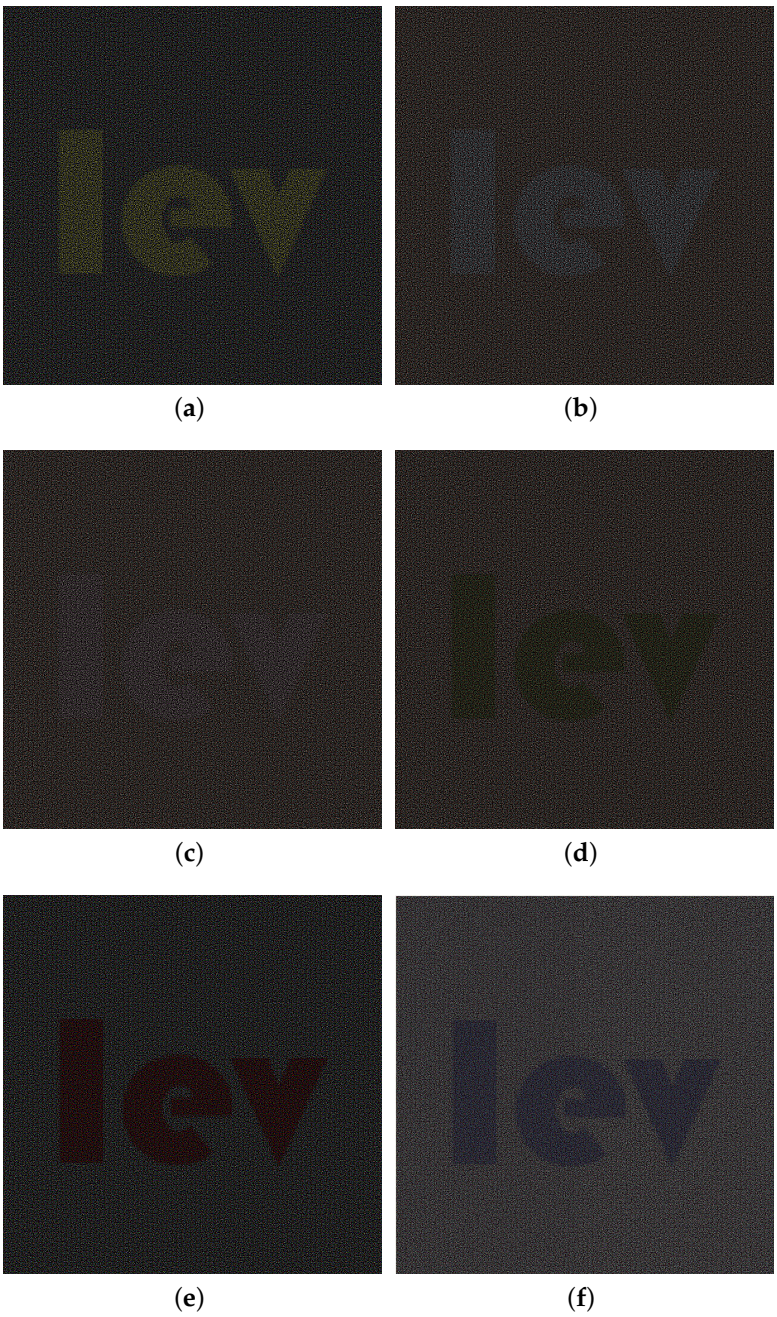


Figure 20. Contrast main observation evaluation chart. (**a**) Yellow. (**b**) Cyan. (**c**) Magenta. (**d**) Green. (**e**) Red. (**f**) Blue.

Lou's algorithm applies to color secret images, but the stacking image has interference from the cover images [29]. This makes the decoding of fine details in the secret image more difficult. In contrast, our algorithm can guarantee a clean stacking result, i.e., no interference from cover images. Both Lou's algorithm and the proposed algorithm scale linearly with the number of pixels in the secret image.

We summarize the above comparison and analysis in Table 2. It is evident that our algorithm is the only one that meets these requirements simultaneously.

Table 2. Comparison with related papers.

Paper	Color Cover Images	No Interference	Meaningful Shares	No Computation
[25]	×	√	×	√
[27]	×	√	×	√
[26]	×	√	√	√
[28]	×	√	√	√
[29]	√	×	√	√
Proposed	√	√	√	√

6. Conclusions

Based on the principle of the VC system, this paper constructs an extended VC system that can embed two secret images on two color cover images. Constraint error diffusion is designed to embed two secret images during the halftoning process. The two secret images are restored by flipping the overlay, and the secret images do not interfere with each other. Experiments show that there is a trade-off between the restoration of secret images and share images. Compared with other related papers, this paper has the better visual quality and good image quality. Our work also demonstrates that joint VC encrypting and halftoning is promising in improving the visual quality of the share images.

Author Contributions: Conceptualization, L.W., B.Y., J.-S.P. and H.-M.Y.; software, L.W., B.Y.; Methodology, J.-S.P., H.-M.Y.; Writing—original draft, L.W., B.Y., J.-S.P. and H.-M.Y.; Writing—review & editing, H.-M.Y. All authors have read and agreed to the published version of the manuscript.

Funding: This work was funded by the the National Natural Science Foundation of China (NSFC) (No. 61272432), Shandong Provincial Natural Science Foundation (No. ZR2014JL044), and MOE (Ministry of Education in China) Project of Humanities and Social Sciences (Project No. 18YJAZH110).

Informed Consent Statement: Not applicable.

Data Availability Statement: The data presented in this study are available on request from the corresponding author. The data are not publicly available due to privacy.

Acknowledgments: The authors would like to thank Molefi Itumeleng Alice for proof-reading the whole manuscript.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Weng, S.; Zhang, C.; Zhang, T.; Chen, K. High capacity reversible data hiding in encrypted images using SIBRW and GCC. *J. Vis. Commun. Image Represent.* **2020**, 102932. [CrossRef]
2. Cai, H.L.; Yan, B.; Pan, J.S.; Ye, J.L. Print-Scan Resistant Two-Level QR Code. *J. Inf. Hiding Multimed. Signal Process.* **2019**, *10*, 300–312.
3. Yan, B.; Chen, N.; Yang, H.M.; Hao, J.J. Local blackness preserving visual cryptography for grayscale secret images. *J. Inf. Hiding Multimed. Signal Process.* **2018**, *9*, 370–382.
4. Naor, M.; Shamir, A. Visual Cryptography. In Proceeding of the Workshop on the Theory and Application of Cryptographic Techniques (EUROCRYPT'94), Perugia, Italy, 9–12 May 1994; pp. 1–12.
5. Sharma, R.G.; Dimri, P.; Garg, H. Visual cryptographic techniques for secret image sharing: A review. *Inf. Syst. Secur.* **2019**, *27*, 241–259. [CrossRef]
6. Yang, C.N.; Laih, C.S. New Colored Visual Secret Sharing Schemes. *Des. Codes Cryptogr.* **2000**, *20*, 325–336. [CrossRef]
7. Mhala, N.C.; Jamal, R.; Pais, A.R. Randomised visual secret sharing scheme for grey-scale and colour images. *IET Image Process.* **2018**, *12*, 422–431. [CrossRef]
8. Chen, S.K. Friendly progressive visual secret sharing using generalized random grids. *Opt. Eng.* **2009**, *48*, 117001. [CrossRef]
9. Hou, Y.C.; Quan, Z.Y. Progressive Visual Cryptography With Unexpanded Shares. *IEEE Trans. Circuits Syst. Video Technol.* **2011**, *21*, 1760–1764. [CrossRef]
10. Wang, R.Z. Region Incrementing Visual Cryptography. *IEEE Signal Process. Lett.* **2009**, *16*, 659–662. [CrossRef]
11. Yang, C.N.; Chung, T.H. A general multi-secret visual cryptography scheme. *Opt. Commun.* **2010**, *283*, 4949–4962. [CrossRef]
12. Dewi, R.; Sari, P.K. The Improvement of Flip (2,2) Visual Cryptography Images Using Two Key Images. *ComTech Comput. Math. Eng. Appl.* **2016**, *7*, 213. [CrossRef]

13. Abdul, W.; Ali, Z.; Ghouzali, S.; Alfawaz, B.; Muhammad, G.; Hossain, M.S. Biometric Security Through Visual Encryption for Fog Edge Computing. *IEEE Access* **2017**, *5*, 5531–5538. [CrossRef]
14. Cimato, S.; Prisco, R.D.; Santis, A.D. Probabilistic Visual Cryptography Schemes. *Comput. J.* **2006**, *49*, 97–107. [CrossRef]
15. Guo, T.; Liu, F.; Wu, C. Threshold visual secret sharing by random grids with improved contrast. *J. Syst. Softw.* **2013**, *86*, 2094–2109. [CrossRef]
16. Nakajima, M.; Yamaguchi, Y. Extended Visual Cryptography for Natural Images. *J. WSCG* **2002**, *10*, 303–310.
17. Liu, F.; Wu, C. Embedded Extended Visual Cryptography Schemes. *IEEE Trans. Inf. Forensics Secur.* **2011**, *6*, 307–322. [CrossRef]
18. Bao, L.; Zhou, Y. Image encryption: Generating visually meaningful encrypted images. *Inf. Sci.* **2015**, *324*, 197–207. [CrossRef]
19. Kanso, A.; Ghebleh, M. An algorithm for encryption of secret images into meaningful images. *Opt. Lasers Eng.* **2017**, *90*, 196–208. [CrossRef]
20. Hsiao, C.Y.; Wang, H.J. Enhancing image quality in Visual Cryptography with colors. In Proceedings of the International Conference on Information Security and Intelligence Control, Yunlin, Taiwan, 14–16 August 2012.
21. Pahuja, S.; Kasana, S.S. Halftone visual cryptography for color images. In Proceedings of the International Conference on Computer, Jaipur, India, 1–2 July 2017; pp. 281–285.
22. Hou, Y.C. Visual cryptography for color images. *Pattern Recognit.* **2003**, *36*, 1619–1629. [CrossRef]
23. Prakash, N.K.; Govindaraju, S. Visual cryptography scheme for color images using halftoning via direct binary search with adaptive search and swap. *Int. J. Comput. Electr. Eng.* **2011**, *3*, 900. [CrossRef]
24. Yan, B.; Xiang, Y.; Hua, G. *Improving Image Quality in Visual Cryptography*; Springer Nature Singapore Pte Ltd.: Singapore, 2019.
25. Lin, S.J.; Chen, S.K.; Lin, J.C. Flip visual cryptography (FVC) with perfect security, conditionally-optimal contrast, and no expansion. *J. Vis. Commun. Image Represent.* **2010**, *21*, 900–916. [CrossRef]
26. Yan, B.; Wang, Y.F.; Song, L.Y.; Yang, H.M. Size-invariant extended visual cryptography with embedded watermark based on error diffusion. *Multimed. Tools Appl.* **2016**, *75*, 11157–11180. [CrossRef]
27. Fang, W.P.; Lin, J.C. Visual cryptography with extra ability of hiding confidential data. *J. Electron. Imaging* **2006**, *15*, 023020. [CrossRef]
28. Wang, Z.; Arce, G.R.; Crescenzo, G.D. Halftone Visual Cryptography Via Error Diffusion. *IEEE Trans. Inf. Forensics Secur.* **2009**, *4*, 383–396. [CrossRef]
29. Lou, D.C.; Chen, H.H.; Wu, H.C.; Tsai, C.S. A novel authenticatable color visual secret sharing scheme using non-expanded meaningful shares. *Displays* **2011**, *32*, 118–134. [CrossRef]