applied sciences

MDPI

Article

Color Visual Secret Sharing for QR Code with Perfect Module Reconstruction

Tao Liu [1], Bin Yan [2] and Jeng-Shyang Pan [1,*]

[1] College of Computer Science and Engineering, Shandong University of Science and Technology, Qingdao 266590, China; taoliu0201@163.com

[2] College of Electronic and Information Engineering, Shandong University of Science and Technology, Qingdao 266590, China; yanbinhit@hotmail.com

* Correspondence: jspan@cc.kuas.edu.tw

Received: 3 October 2019; Accepted: 29 October 2019; Published: 2 November 2019

Abstract: Visual secret sharing is a secret sharing scheme where the decryption requires no computation. It has found many applications in online transaction security, privacy protection, and bar code security, etc. Recently, researches have indicated that combining visual secret sharing with the widely used Quick Response code may provide additional security mechanism to online transaction. However, current methods are either pixel-based, which requires high computational complexity or module-based, which sacrifices error correction capability of the original Quick Response code. Designing module-based visual secret sharing for the Quick Response code without sacrificing error correction capability is a challenging problem. To solve this problem, this paper proposes a (3, 3)-threshold visual secret sharing for Quick Response code scheme that fully explores the extra freedom provided by color visual secret sharing and color stacking. The binary secret Quick Response code is encoded into color shares. By stacking all the three shares, a binary color Quick Response code can be reconstructed. After the inherent pre-processing steps in a standard Quick Response code decoder, the original binary secret Quick Response code can be completely reconstructed. Thus, the original error correction capability of the Quick Response code is fully preserved. Theoretical analysis shows that the visual secret sharing for Quick Response code is secure under the condition that the computational device available to the attacker is limited to a decoder for standard Quick Response code. Experimental results verify that the secret Quick Response code cannot be reconstructed from just one share or any two shares. However, it can be 100% reconstructed once the three shares are stacked. The proposed visual secret sharing for Quick Response code is module-based, and it does not sacrifice the error correction capability. Furthermore, No extra pre-processing steps other than the standard Quick Response code decoder are required.

Keywords: visual secret sharing; Quick Response code; color stacking

1. Introduction

QR (Quick Response) code has been widely used since 1994 when it was invented by the DENSO WAVE corporation of Japan. Now it becomes an integral part of our everyday lives and makes our lives more and more convenient. QR code can be used in information storage, website access, online transaction and online banking etc.

These applications call for protection of data security and privacy. However, QR code itself does not provide such mechanism. It can be scanned and read by anyone using a standard QR code decoder. To ensure security, an extra mechanism should be designed and incorporated into the standard QR code, such as encryption, authentication, privacy protection, hiding, and secret sharing. Encryption, authentication and privacy protection can be ensured by resorting to classical

cryptography techniques [1–3], such as private key cryptography, message authentication code, public key cryptography [4,5], and digital signature, etc. Data hiding and secret sharing are new techniques for multimedia security, but can also be tailored to ensure QR code security.

Lin et al. hided secret data into QR code by using error correction capability [6]. Huang et al. proposed a data embedding mechanism to hide secret data into QR code by using Reed-Solomon code and turtle shell matrix [7]. Furthermore, reversible data hiding (RDH) can be combined with QR code to ensure data security and data recovery. Huang et al. proposed a RDH algorithm for QR code [8]. The background image covered by QR code is embedded into other regions of the background image using RDH [9–16]. Once scanned, the QR code can be removed and the background image can be partially recovered.

VSS (Visual Secret Sharing) was first proposed by Naor and Shamir in 1994 [17]. The main idea of VSS is that the secret image can be divided into multiple share images, from which the secret images can be recovered by stacking or logical "OR" operation. The idea of VSS can be combined with many multimedia security applications [18–22], such as QR code security. QR code image poses some special features than the natural image or binary image that are used as input to ordinary VSS algorithm. For example, the observer of a QR code is not the human vision sytem (HVS), but a QR code decoder [23]. These features bring new ingredients into the VSS algorithm design.

Fang [24] applied VSS to share a secret QR code. Fang's algorithm can completely reconstruct the secret QR code from shares. However, this algorithm is pixel-based, i.e., each secret pixel of the QR code image is encrypted independently. The computational complexity is proportional to the number of pixels in a QR code, which is significantly larger than module-based processing. However, if this algorithm is applied directly to each module of the QR code, the error probability of the reconstructed module is about 50%, which exceeds the ECC of QR code.

Lin et al. also proposed VSS algorithm using QR code as cover images [25]. The secret data was split into n shares using a random grid approach. Each share is then embedded into one QR code such that the modification of each QR code is within its error correction capability. However, this algorithm uses QR codes as cover images, not as the secret image. Furthermore, it needs computation when reconstructing the secret image.

Recently, Chow et al. proposed QRCSS (QR Code Secret Sharing) algorithm to achieve QR code secret sharing [26]. The distortion between the reconstructed QR code and the secret QR code can be controlled such that the reconstructed QR code is readable by a QR code decoder. To achieve this goal, QRCSS utilizes the error correction capability of the QR code. However, this damages the error tolerance of the QR code, leaving smaller margin for error correction. Furthermore, QRCSS needs to get all the n shares in order to to reconstruct the secret QR code. To relax this constraint, Cheng et al. proposed an improved algorithm that needs only k ($k < n$) shares in order to reconstruct the secret QR code [27].

In summary, to facilitate fast processing and to preserve the error correction code (ECC) capability, a module-based VSS for QR code with perfect reconstruction should be designed. However, for black and white (B/W) QR code, achieving these two goals simultaneously is a challenging problem.

To design QR code VSS algorithm with module-based processing and preserving ECC, this paper proposes VSS-QR (VSS for QR code) system by exploring the extra freedom provided by color QR code. The secret QR code can be any standard B/W QR code. VSS-QR can hide this secret QR code into three meaningless color shares. At the decoding side, a standard QR code decoder can perfectly reconstruct the original secret QR code, using only the standard pre-processing in QR code decoder.

VSS-QR is different from the ordinary color VC algorithm in [28], which is designed for natural color image and the secret image cannot be perfected reconstructed. The contributions of the proposed algorithm can be summarized as follows.

1. Compared with Fang's algorithm [24], the proposed algorithm is module-based. The basic processing unit of VSS-QR is a module in QR code, not pixel. So, the computational complexity can be significantly reduced.

2. Compared with Lin's algorithm [25] and Chow's algorithm [26], the proposed VSS-QR does not sacrifice ECC capability of the QR code, i.e., the secret QR code can be completely reconstructed at decoder.

2. Related Background

This section briefly reviews some related backgrounds, including the structure and processing steps for QR code (Section 2.1), the RGB color space and color stacking (Section 2.2), and requirements for VSS-QR scheme (Section 2.3).

2.1. QR Code

QR code is a two-dimensional array composed of light and dark modules. It has 40 versions corresponding to different sizes and payload. Version 1 is made up of 21 × 21 modules. The edge length of each subsequent version is four modules longer than that of the previous version. For each version, QR code has four error correction levels: L, M, Q and H. Error correction level L can correct up to approximately 7% codeword errors. Similarly, level M, Q, H can correct roughly up to 15%, 25%, and 30% of codeword errors, respectively.

As shown in Figure 1, the structure of a typical QR code can be divided into two regions: function patterns and encoding region [23]. The function of finder pattern is to help the decoder to locate the region of a QR code image. The function of timing pattern can enable the symbol density and version to be determined and provide datum positions for determining module coordinates. Every data codeword is made up of eight modules. The same is true for every error correction codeword.

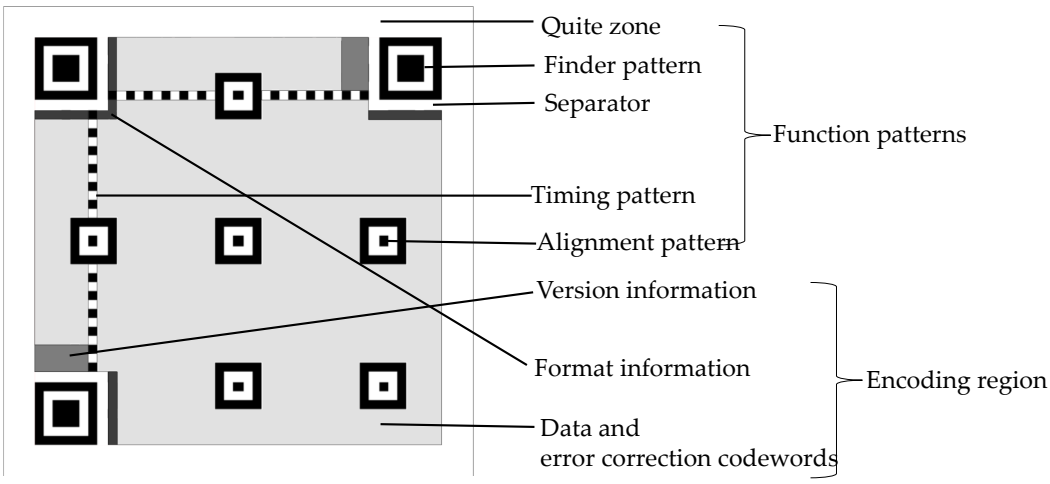

Figure 1. Structure of a typical QR code (Version 7).

In the *encoding* stage, the input data (such as text or numbers) is encoded into a QR code image, which can be described by the following steps:

1. Determine the type of data encoding according to the type of the input data.
2. Convert data characters into bit stream.
3. Convert the bit stream into codewords and assemble codewords into blocks for ECC encoding.
4. Error correction codeword encoding, which calculates error correction codewords for each block.
5. Place the codeword modules into the QR code matrix, along with the finder pattern, separator, timing pattern, and alignment pattern.
6. Masking: XOR (exclusive OR) the QR code matrix with a selected masking pattern to balance the B/W modules in encoding region.
7. Generate and place the format information and version information.

In the *decoding* stage, a captured color QR code image is processed and analyzed to extract the embedded data. This can be divided into the following steps:

1. Convert color QR code image into a binary image. The brightness component of the color QR code image is extracted and compared to a global/local threshold to get binary image.
2. Use the binary image in last step to determine a sampling grid for modules and decode each module into bits: the dark and light modules are identified as "1" and "0", respectively.
3. Read the format and version information to determine the version of the symbol.
4. Eliminate mask by "XOR" operation.
5. Use error correction codeword decoder to obtain data codewords.
6. Decode the data codewords to obtain the embedded data (such as text or numbers).

2.2. RGB Color Space and Color Stacking

2.2.1. RGB Color Space

The proposed VSS-QR is mainly designed by using RGB color space [29], so we review it briefly. Compared with CMY color space, RGB color space has the following features for VSS application.

1. RGB color model has no color darkening while CMY color model has [30].
2. RGB model is suitable for screen displaying. Considering the trend in QR code usage nowadays that more and more QR code are displayed on screen (computer monitor, screen of a smart phone or a tablet), VSS-QR using RGB color space complies with this trend.

RGB color space is an additive color system, which is commonly used when presenting colors electronically. As shown in Figure 2, when mixing any two colors of red, green and blue, a brighter secondary color can be obtained, such as cyan, magenta and yellow. When mixing the same amount of red, green and blue, the white color should be obtained.

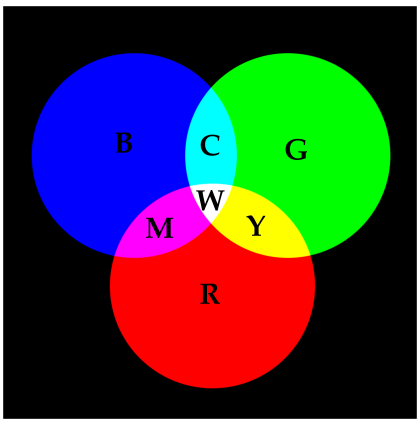

Figure 2. Color mixing in RGB additive color space.

In RGB color space, red, green and blue colors are used as three basis colors. Any color in this space can be represented as linear combination of these three colors. Let X represents a given color in RGB space, then X can be expressed by

$$X = r \times R + g \times G + b \times B, \tag{1}$$

where R, G and B represents the red, green and blue colors, respectively. Thus, any color in a RGB color space can be represented as a vector $[r,\ g,\ b]^T$. For convenience, we use boldface San-Serif font to

denote some commonly used color vectors. The color vectors corresponding to red, green, blue, cyan, magenta, yellow, white and black are denoted as:

$$\begin{aligned} &\mathsf{R} = [1\ 0\ 0]^T,\ \mathsf{G} = [0\ 1\ 0]^T,\ \mathsf{B} = [0\ 0\ 1]^T,\ \mathsf{C} = [0\ 1\ 1]^T,\ \mathsf{M} = [1\ 0\ 1]^T, \\ &\mathsf{Y} = [1\ 1\ 0]^T,\ \mathsf{W} = [1\ 1\ 1]^T,\ \mathsf{K} = [0\ 0\ 0]^T. \end{aligned} \tag{2}$$

2.2.2. Color Stacking in RGB Space

In a typical VSS system, the decoder needs to stack shares in order to decode the secret bits. Such stacking is equivalent to logic OR operation if shares are assumed to be printed on transparencies. Let $a \sqcap d$ represents the logic OR operation between two bits a and d, then we have $0 \sqcap 0 \rightarrow 0$, $0 \sqcap 1 \rightarrow 0$, $1 \sqcap 1 \rightarrow 1$, $1 \sqcap 0 \rightarrow 0$. For any two colors from the eight colors $\mathcal{C} = \{\mathsf{R},\ \mathsf{G},\ \mathsf{B},\ \mathsf{C}, \mathsf{M},\ \mathsf{Y},\ \mathsf{W},\ \mathsf{K}\}$, the stacking operation can also be represented as component-wise OR between color vectors. Let $\mathbf{a} = [a_r,\ a_g,\ a_b]^T$ and $\mathbf{d} = [d_r,\ d_g,\ d_b]^T$ be two colors in $\mathcal{C}$, then the stacking of $\mathbf{a}$ and $\mathbf{d}$ is

$$\mathbf{a} \sqcap \mathbf{d} = [a_r,\ a_g,\ a_b]^T \sqcap [d_r,\ d_g,\ d_b]^T = [a_r \sqcap d_r,\ a_g \sqcap d_g,\ a_b \sqcap d_b]^T. \tag{3}$$

For example, the stacking of $\mathsf{R} \sqcap \mathsf{B}$ is

$$\mathsf{R} \sqcap \mathsf{B} = [1\ 0\ 0]^T \sqcap [0\ 0\ 1]^T == [1\ 0\ 1]^T = \mathsf{M}. \tag{4}$$

The stacking operation can be extended to more than two shares. Some examples of stacking of three shares are illustrated in Table 1.

Table 1. Color stacking of three shares.

Color 1 ($\mathbf{c_1}$)	**Color 2** ($\mathbf{c_2}$)	**Color 3** ($\mathbf{c_3}$)	X ($\mathbf{c_1} \sqcap \mathbf{c_2} \sqcap \mathbf{c_3}$)
R	G	B	W
K	G	B	C
R	K	B	M
R	G	K	Y
C	M	Y	W
C	K	Y	W
C	B	K	C

2.3. Requirements for VSS-QR

The VSS-QR has a different setting as ordinary VSS because the decoders are different. For ordinary VC, the decoder consists of a stacking operation and a HVS. In contrast, for VSS-QR, the decoder consists of stacking operation and a QR code decoder.

Taking the $(n,\ n)$-threshold scheme as an example, ordinary VC should satisfy two conditions. (1) Contrast condition: If all n shares are stacked, then the secret image should be revealed with sufficient contrast. (2) Security condition: If less than n shares are collected, no information about the secret bits should be revealed. For a VSS-QR scheme, in order to define the contrast condition and security condition, we must determine the computational ability of the attacker and define the meaning of "contrast" for a QR code decoder.

Security should be defined under an assumption of the ability of a potential attacker. This ability is quantified by the computational resources the attacker may have. In this paper, we assume that the attacker has a standard QR code decoder which may help him to reveal the embedded data in a QR code.

Contrast in VSS-QR is different from contrast in ordinary VSS. In ordinary VSS, one focuses on the difference between white region and black region. The larger the difference, the better the contrast. However, this contrast is defined for HVS, not for a QR code decoder. Using a QR code decoder as an

observer, if the number of codeword errors is within ECC capability, then the "contrast" is acceptable for the QR code decoder. Otherwise, the "contrast" is not large enough for QR code decoder. So, here 'contrast' is defined as correct decoding.

In summary, a VSS-QR should satisfy the following conditions.

1. *Contrast condition:* If all n shares are collected and stacked, then the recovered secret QR code should be decoded correctly by a standard QR code decoder.
2. *Security condition:* If k shares $(k < n)$ are collected, then the secret QR code should not be decoded by stacking any combination of these k shares.

3. The Proposed VSS-QR

The overall block diagram of the proposed scheme is shown in Figure 3. It consists of an encoder and a decoder. The secret QR code $\mathbf{H}$ is split into three shares by a VSS-QR algorithm. These three shares are transmitted to the receiver. At the receiving end, the receiver stacks all the three shares $\mathbf{S}_1$, $\mathbf{S}_2$, $\mathbf{S}_3$ to generate a color QR code $\mathbf{T}$. Using a standard QR code decoder, the color QR code can be converted into a B/W QR code $\mathbf{T}'$, which then can be decoded as an ordinary QR code. The encoder is described in Section 3.1 and the decoder is described in Section 3.2. The proposed algorithm is a $(3, 3)$-threshold VSS-QR, which can be extended to (k, n)-threshold VSS-QR.

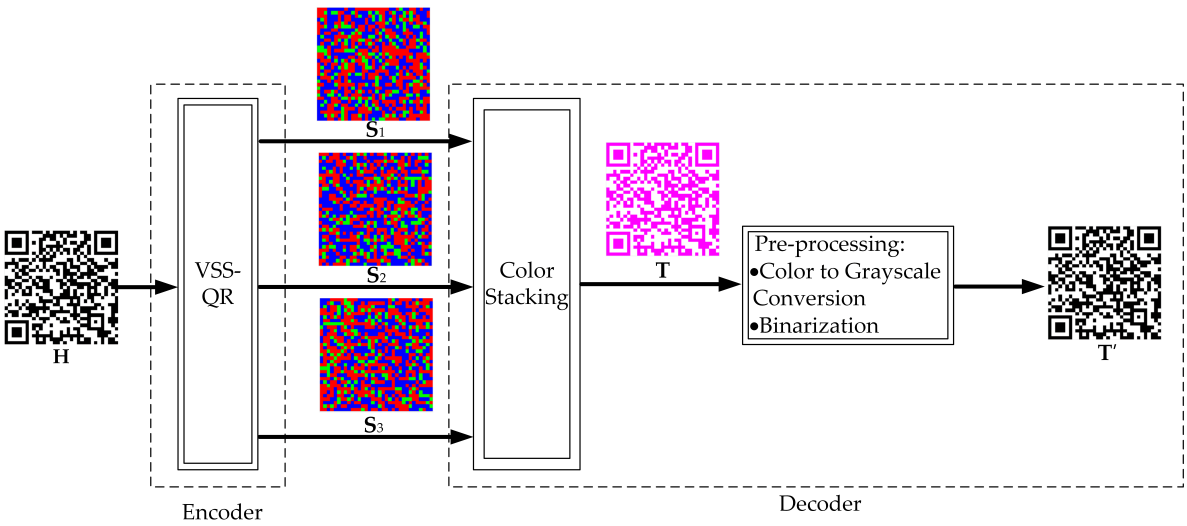

Figure 3. The overall block diagram.

3.1. VSS-QR Encoder

Given a secret QR code $\mathbf{H}$, the encoder generates three shares $\mathbf{S}_1$, $\mathbf{S}_2$, $\mathbf{S}_3$. To ensure perfect reconstruction of $\mathbf{H}$ at the decoder, we need to design the set of colors for encoding B/W secret modules. Let $\mathcal{E}_1$ and $\mathcal{E}_0$ be the two sets of colors for W/B secret modules, respectively. We intend to ensure that, a white secret module is constructed as white module and a black secret module is constructed as a module with a secondary color (such as cyan, magenta or yellow). To this end, we design $\mathcal{E}_1$ and $\mathcal{E}_0$ as follows. The set $\mathcal{E}_1$ is designed as $\mathcal{E}_1 = \{\mathsf{R}, \mathsf{G}, \mathsf{B}\}$ since all three primary colors are needed in order to generate a white module. To design $\mathcal{E}_0$, we need to determine the reconstructed color for black secret module. For example, if the black secret module should be reconstructed as magenta, then we set $\mathcal{E}_0 = \{\mathsf{R}, \mathsf{B}\}$. In summary, let $\mathbf{H}(x, y)$ and $\mathbf{T}(x, y)$ be the secret module and reconstructed module at position (x, y), respectively. Then we set

$$\mathcal{E}_0 = \begin{cases} \{\mathsf{G},\ \mathsf{B}\}\,, \text{ if } \mathbf{T}(x,\ y) = \mathsf{C}, \\ \{\mathsf{R},\ \mathsf{B}\}\,, \text{ if } \mathbf{T}(x,\ y) = \mathsf{M}, \\ \{\mathsf{R},\ \mathsf{G}\}\,, \text{ if } \mathbf{T}(x,\ y) = \mathsf{Y}, \end{cases} \tag{5}$$

if $\mathbf{H}(x,\ y) = 0$. If the color for the reconstructed module is not specified, then one may randomly select $\mathcal{E}_0$ from the set $\{\{\mathsf{G},\ \mathsf{B}\}\,,\ \{\mathsf{R},\ \mathsf{B}\}\,,\ \{\mathsf{R},\ \mathsf{G}\}\}$. Once designed, the two sets $\mathcal{E}_0$ and $\mathcal{E}_1$ remain fixed during the encoding process.

For clarity, let us focus on only one module in the secret QR code $\mathbf{H}$ and the reconstructed QR code $\mathbf{T}$. To share a black module $\mathbf{H}(x,\ y) = 0$, we must ensure that the two colors in $\mathcal{E}_0$ appear at least once in the three shares $\mathbf{S}_1(x,\ y)$, $\mathbf{S}_2(x,\ y)$, $\mathbf{S}_3(x,\ y)$. Let $p \sim \mathcal{P}$ denotes random selecting an element from the set $\mathcal{P}$ and assigning it to p, and let $\mathcal{P} \backslash \mathcal{Q}$ denotes the difference set between two sets $\mathcal{P}$ and $\mathcal{Q}$. Then we have

$$\mathbf{S}_1(x,\ y) \sim \mathcal{E}_0, \tag{6}$$
$$\mathbf{S}_2(x,\ y) \sim \mathcal{E}_0, \tag{7}$$
$$\mathbf{S}_3(x,\ y) \sim \mathcal{E}_0 \backslash \{\mathbf{S}_1(x,\ y)\}\,. \tag{8}$$

This operation can ensure that the stacking result of the three shares is a secondary color, i.e., $\sqcap_{i=1}^{3} \mathbf{S}_i(x,\ y) \in \{\mathsf{C},\ \mathsf{M},\ \mathsf{Y}\}$.

To share a white module $\mathbf{H}(x,\ y) = 1$, we must ensure that the three colors in $\mathcal{E}_1$ appear at least once in the three shares $\mathbf{S}_1(x,\ y)$, $\mathbf{S}_2(x,\ y)$, $\mathbf{S}_3(x,\ y)$. We may use the following operations:

$$\mathbf{S}_1(x,\ y) \sim \mathcal{E}_1, \tag{9}$$
$$\mathbf{S}_2(x,\ y) \sim \mathcal{E}_1 \backslash \{\mathbf{S}_1(x,\ y)\}\,, \tag{10}$$
$$\mathbf{S}_3(x,\ y) \sim \mathcal{E}_1 \backslash \{\mathbf{S}_1(x,\ y), \mathbf{S}_2(x,\ y)\}\,. \tag{11}$$

These operations can ensure that the stacking result of the three shares is white color, i.e., $\sqcap_{i=1}^{3} \mathbf{S}_i(x,\ y) = \mathsf{W}$.

For example, if we select $\mathcal{E}_0 = \{\mathsf{R},\ \mathsf{B}\}$ and $\mathcal{E}_1 = \{\mathsf{R},\ \mathsf{G},\ \mathsf{B}\}$, then for $\mathbf{H}(x,\ y) = 0$, we may get $\mathbf{S}_1(x,\ y) = \mathsf{R}$, $\mathbf{S}_2(x,\ y) = \mathsf{R}$, $\mathbf{S}_3(x,\ y) = \mathsf{B}$ and $\sqcap_{i=1}^{3} \mathbf{S}_i(x,\ y) = \mathsf{M}$. For $\mathbf{H}(x,\ y) = 1$, we may get $\mathbf{S}_1(x,\ y) = \mathsf{B}$, $\mathbf{S}_2(x,\ y) = \mathsf{R}$, $\mathbf{S}_3(x,\ y) = \mathsf{G}$ and $\sqcap_{i=1}^{3} \mathbf{S}_i(x,\ y) = \mathsf{W}$.

By applying the above encoding algorithm to each module of $\mathbf{H}$ sequentially, we get three shares, each having $D \times D$ modules, where D is the number of modules along each row/column of the secret QR code. For QR code with version V, D can be calculated as:

$$D = 17 + 4 \times V. \tag{12}$$

To render the shares as images, we must determine the size of each module. Let the size of each square module be a $L \times L$ pixels, then each share image has $L \cdot D \times L \cdot D$ pixels. For convenience, we use the same letter for QR code image with module indexing and QR code image with pixel indexing, where brackets are used for pixel indexing and parentheses are used for module indexing. For example, the QR code image $\mathbf{H}$ is written as $\mathbf{H}(x,\ y)$ when accessing the $(x,\ y)$ module, but is written as $\mathbf{H}[i,\ j]$ when accessing the $[i,\ j]$ pixel, where $1 \le x, y \le D$ and $1 \le i, j \le LD$. The image rendering process can be expressed as:

$$\mathbf{H}[i,\ j] = \mathbf{H}(x,\ y), \tag{13}$$

for $i \in [(x-1)L + 1,\ xL]\,,\ j \in [(y-1)L + 1,\ yL]$.

The above encoding algorithm is summarized in Algorithm 1.

Algorithm 1 (3,3)-threshold VSS-QR algorithm

Input:
Secret QR code $\mathbf{H}$
Output:
Three shares: $\mathbf{S}_1,\ \mathbf{S}_2,\ \mathbf{S}_3$.

Determine the two sets $\mathcal{E}_0$ and $\mathcal{E}_1$:
$\mathcal{E}_0 \sim \{\{\mathsf{G},\ \mathsf{B}\},\ \{\mathsf{R},\ \mathsf{B}\},\{\mathsf{R},\ \mathsf{G}\}\}, \mathcal{E}_1 = \{\mathsf{R},\ \mathsf{G},\ \mathsf{B}\}$
for x from 1 to D **do**
  for y from 1 to D **do**
    if $\mathbf{H}(x,\ y) = 0$ **then**
      $\mathbf{S}_1(x,\ y) \sim \mathcal{E}_0$,
      $\mathbf{S}_2(x,\ y) \sim \mathcal{E}_0$,
      $\mathbf{S}_3(x,\ y) \sim \mathcal{E}_0 \backslash \{\mathbf{S}_1(x,\ y)\}$.
    else
      $\mathbf{S}_1(x,\ y) \sim \mathcal{E}_1$,
      $\mathbf{S}_2(x,\ y) \sim \mathcal{E}_1 \backslash \{\mathbf{S}_1(x,\ y)\}$,
      $\mathbf{S}_3(x,\ y) \sim \mathcal{E}_1 \backslash \{\mathbf{S}_1(x,\ y), \mathbf{S}_2(x,\ y)\}$.
    end if
    $\mathbf{S}_\ell[i,\ j] = \mathbf{H}(x,\ y)$, for $i \in [(x-1)L+1,\ xL], j \in [(y-1)L+1,\ yL], \ell = 1,\ 2,\ 3$.
  end for
end for

3.2. VSS-QR Decoder

After receiving all the three shares, the decoder stacks them to get the reconstructed color QR code $\mathbf{T}$ (Figure 3). Fortunately, this color QR code is directly recognizable by a standard QR code decoder. A QR code decoder first pre-process $\mathbf{T}$ to generate a binary QR code $\mathbf{T}'$, which includes color to grayscale conversion and binarization. Even though these pre-processing steps are assembled into a QR code decoder, we describe them here since they are essential for understanding the perfect reconstruction feature of VSS-QR.

3.2.1. Color Stacking

To reconstruct the secret QR code, the decoder stacks the three shares pixel by pixel:

$$\mathbf{T}[i,\ j] = \sqcap_{\ell=1}^{3} \mathbf{S}_\ell[i,\ j]. \tag{14}$$

for $1 \le i, j \le LD$. One illustrating result is shown in Figure 3.

3.2.2. Color to Grayscale Conversion

The conversion from color to grayscale can be accomplished by a standard QR code decoder. However, the QR code standard does not specify how this conversion should be done. Different implementations are allowed. One possible solution is to convert the RGB space to HSI (Hue, Saturation and Intensity) color space, and then retain only the intensity component. Here we use "weighted average method" that assigns different weights to different color channels [31]. Using weighted average, the perceptually more important green channel are assigned with the largest weight. The weight vector is $\mathbf{w}$=[0.299, 0.587, 0.114]. So, the grayscale image can be calculated as:

$$\begin{aligned} \hat{\mathbf{T}}[i,\ j] &= \mathbf{w}^T \cdot \mathbf{T}[i,j] \\ &= 0.299 \times T_r[i,\ j] + 0.587 \times T_g[i,\ j] + 0.114 \times T_b[i,\ j], \end{aligned} \tag{15}$$

for all $1 \le i,\ j \le LD$, where $\mathbf{T}[i,\ j] = \left[T_r[i,\ j], T_g[i,\ j], T_b[i,\ j]\right]^T$.

3.2.3. Binarization

The grayscale QR code image needs to be binarized to facilitate QR code localization and module sampling in later steps. An advantage of QR code binarization is that we know in *a priori* that there are two classes of pixels in clean image, black pixels and white pixels. Thus the grayscale image $\hat{\mathbf{T}}$ should have two peaks in its histogram. With this knowledge, we may utilize Otsu's algorithm to find the best threshold [32].

Let the histogram of the image $\hat{\mathbf{T}}$ be $h(\hat{g})$, where $0 \leq \hat{g} \leq 255$. Given a trail threshold τ, all pixels can be classified as either background pixels or foreground pixels. Let the mean for background, foreground and entire image be calculated as:

$$\mu_b = \frac{\sum_{\hat{g}=0}^{\tau} \hat{g} \cdot h(\hat{g})}{N_b}, \ \mu_f = \frac{\sum_{\hat{g}=\tau+1}^{255} \hat{g} \cdot h(\hat{g})}{N_f}, \ \mu = \frac{\sum_{\hat{g}=0}^{255} h(\hat{g})}{N_b + N_f}. \tag{16}$$

where $N_b = \sum_{\hat{g}=0}^{\tau} h(\hat{g})$ and $N_f = \sum_{\tau+1}^{255} h(\hat{g})$ are the number of background pixels and number of foreground pixels, respectively. Let $N = N_b + N_f$, then the intra-class variance can then be calculated as:

$$\sigma^2(\tau) = \frac{N_b}{N} (\mu_b - \mu)^2 + \frac{N_f}{N} \left(\mu_f - \mu\right)^2. \tag{17}$$

The optimal $\tau^\star$ can be determined by maximizing this intra-class variance $\sigma^2(\tau)$. Using $\tau^\star$, we obtain the binary QR code image as:

$$\mathbf{T}'[i, j] = \begin{cases} 1, & \text{if } \hat{\mathbf{T}}[i, j] \geq \tau^\star, \\ 0, & \text{if } \hat{\mathbf{T}}[i, j] < \tau^\star. \end{cases} \tag{18}$$

3.2.4. Decoding by Standard QR Code Scanner

After the above pre-processing steps, we get a B/W QR code image $\mathbf{T}'$, which can be decoded by a standard decoder such as ZXing library [33].

3.3. Computational Complexity

The computational complexity of the proposed VSS-QR is proportional to the number of modules since the encoding is module-based. To be specific, the computational complexity is $\mathcal{O}(D^2)$. In contrast, Fang's algorithm is pixel-based, so the computational complexity is $\mathcal{O}(L^2 D^2)$.

3.4. Timing Analysis

The basic processing unit of VSS-QR is a module, not pixel. The module is the component unit of QR code. The larger the version of the QR code, the more modules it has. Therefore, the size of the version is the main factor affecting the execution time of the algorithm. The execution time of the algorithm is gradually increased when the version is gradually increased.

4. Security Analysis

In this section, we show that the $(3, 3)$-threshold VSS-QR satisfies the security requirement as listed in Section 2.3. The QR code is assumed to be QR code 4-H, where the version of the QR code is 4 and the error correction level is H. This analysis can be extended to (k, n)-threshold VSS-QR and other versions of QR code as well.

Since the encoding for black secret modules and white secret modules are different, so we define Ω_0 and Ω_1 as the sets of module positions corresponding to black modules and white modules, respectively, i.e.,

$$\Omega_0 = \{(x, y) | \text{ such that } \mathbf{H}(x, y) = 0\}, \tag{19}$$
$$\Omega_1 = \{(x, y) | \text{ such that } \mathbf{H}(x, y) = 1\}. \tag{20}$$

Next, we show that, after obtaining one or two shares, the attacker is unable to recover the secret QR code and hence is unable to read the embedded message. Without loss of generality, we assume that the encoder choose $\mathcal{E}_0 = \{\mathsf{R}, \mathsf{B}\}$ and $\mathcal{E}_1 = \{\mathsf{R}, \mathsf{G}, \mathsf{B}\}$ as in Section 3.1.

4.1. Decoding from One Share

Given only one share, we show that the probability of codeword error is far beyond the ECC capability of QR code version 4-H.

The share image consists of color module, so a conversion from color module to B/W module is necessary. This problem is referred as *module recovery* and can be stated as follows. A secret QR code $\mathbf{H}$ is encoded into three shares $\mathbf{S}_1$, $\mathbf{S}_2$, $\mathbf{S}_3$. Given only one share $\mathbf{S} \in \{\mathbf{S}_1, \mathbf{S}_2, \mathbf{S}_3\}$, where $\mathbf{S}(x, y) \in \{\mathsf{R}, \mathsf{G}, \mathsf{B}\}$, determine a binary QR code $\hat{\mathbf{H}}(x, y) \in \{0, 1\}$ to minimize the probability that $\hat{\mathbf{H}}(x, y) \neq \mathbf{H}(x, y)$. We define the module recovery error as

$$e_m \triangleq \left\{\hat{\mathbf{H}}(x, y) \neq \mathbf{H}(x, y)\right\}. \tag{21}$$

To minimize the probability of module recovery error $\Pr(e_m)$, the optimal detector is a MAP (maximum a posteriori probability) detector under the condition of equal probability for $\mathbf{H}(x, y) \in \{0, 1\}$ [34].

To calculate the a posteriori probabilities, recall that $\mathcal{E}_0 = \{\mathsf{R}, \mathsf{B}\}$ and $\mathcal{E}_1 = \{\mathsf{R}, \mathsf{G}, \mathsf{B}\}$. So, given a blue module in $\mathbf{S}$, we have

$$\Pr\left(\mathbf{H}(x, y) = 0 | \mathbf{S}(x, y) = \mathsf{G}\right) = 0, \tag{22}$$
$$\Pr\left(\mathbf{H}(x, y) = 1 | \mathbf{S}(x, y) = \mathsf{G}\right) = 1. \tag{23}$$

To calculate other a posteriori probabilities, we need $\Pr\left(\mathbf{H}(x, y) = t_1\right)$, where $t_1 \in \{0, 1\}$, and $\Pr\left(\mathbf{S}(x, y) = t_2\right)$, where $t_2 \in \{\mathsf{R}, \mathsf{G}, \mathsf{B}\}$. For the secret QR code, it is usually assumed that $\Pr\left(\mathbf{H}(x, y) = 0\right) = \Pr\left(\mathbf{H}(x, y) = 1\right) = 1/2$, because the masking operation balances the number of B/W modules. For the share image $\mathbf{S}$, we can obtain:

$$\Pr\left(\mathbf{S}(x, y) = \mathsf{R}\right) = \frac{1}{2} \cdot \frac{1}{2} + \frac{1}{2} \cdot \frac{1}{3} = \frac{5}{12}, \tag{24}$$
$$\Pr\left(\mathbf{S}(x, y) = \mathsf{G}\right) = \frac{1}{2} \cdot \frac{1}{3} = \frac{1}{6}, \tag{25}$$
$$\Pr\left(\mathbf{S}(x, y) = \mathsf{B}\right) = \frac{1}{2} \cdot \frac{1}{2} + \frac{1}{2} \cdot \frac{1}{3} = \frac{5}{12}. \tag{26}$$

The a posteriori probability then can be determined from the Bayes rule:

$$\Pr\left(\mathbf{H}(x, y) = 0 | \mathbf{S}(x, y) = \mathsf{R}\right) = \frac{\frac{1}{2} \times \frac{1}{2}}{\frac{1}{2} \times \frac{1}{2} + \frac{1}{2} \times \frac{1}{3}} = \frac{3}{5}. \tag{27}$$

Similarly, we get $\Pr\left(\mathbf{H}(x, y) = 1 | \mathbf{S}(x, y) = \mathsf{R}\right) = 2/5$, $\Pr\left(\mathbf{H}(x, y) = 0 | \mathbf{S}(x, y) = \mathsf{B}\right) = 3/5$ and $\Pr\left(\mathbf{H}(x, y) = 1 | \mathbf{S}(x, y) = \mathsf{B}\right) = 2/5$, which are summarized in Table 2.

Table 2. Color probability diagram of a share.

Color(c)	$\mathbf{Pr}(\mathbf{S}(x,\ y) = \mathbf{c})$	$\mathbf{Pr}(\mathbf{H}(x,\ y) = \mathbf{0}\vert\mathbf{S}(x,\ y) = \mathbf{c})$	$\mathbf{Pr}(\mathbf{H}(x,\ y) = \mathbf{1}\vert\mathbf{S}(x,\ y) = \mathbf{c})$
	$\frac{5}{12}$	$\frac{3}{5}$	$\frac{2}{5}$
	$\frac{1}{6}$	0	1
	$\frac{5}{12}$	$\frac{3}{5}$	$\frac{2}{5}$

Based on the above a posteriori probabilities, the MAP module detector decodes

$$\hat{\mathbf{H}}(x,\ y) = \begin{cases} 0, & \text{if } \mathbf{S}(x,\ y) \in \{\mathsf{R},\ \mathsf{B}\}, \\ 1, & \text{if } \mathbf{S}(x,\ y) = \mathsf{G}. \end{cases} \tag{28}$$

The probability of module recovery error can be calculated as:

$$\begin{aligned} \Pr(e_m) &= \Pr\left\{\hat{\mathbf{H}}(x,\ y) \neq \mathbf{H}(x,\ y)\right\} \\ &= 2 \cdot \frac{5}{12} \cdot \Pr\left(\hat{\mathbf{H}}(x,\ y) = 0 \vert \mathbf{H}(x,\ y) = 1\right) \\ &= \frac{1}{3}. \end{aligned} \tag{29}$$

Since each codeword consists of 8 modules, the probability of codeword error e_c is

$$\Pr(e_c) = 1 - \prod_{t=1}^{8} [1 - \Pr(e_m(t))] = 1 - \left(\frac{2}{3}\right)^8 = 0.961. \tag{30}$$

where $e_m(t)$ denotes error of the t-th module in the codeword. For QR code (4-H), there are four blocks. For each block, the Reed-Solomon code is $(\hat{c},\ \hat{k},\ \hat{r}) = (25,\ 9,\ 8)$ [23], where $\hat{c}$ is the total number of codewords in a block, $\hat{k}$ the number of data codewords, and $\hat{r}$ is the ECC capability. The QR code is successfully decoded if the number of codeword errors is less than 8 in each block. So, the probability of correct decoding should be:

$$\prod_{\ell=1}^{4} \sum_{q=1}^{8} \binom{25}{q} \Pr(e_c)^q \left[1 - \Pr(e_c)\right]^{25-q} = 6.41 \times 10^{-73}. \tag{31}$$

According to Equation (31), the attacker needs to try on approximately 10^{73} shares in order to get a correct decoding, which is almost impossible, by considering the fact that the total number of atoms in the universe is around 10^{80}.

In summary, given only one share, it is computational infeasible to decode the secret QR code correctly.

4.2. Decoding from Two Shares

Given any two shares, we show that the probability of codeword error is also far beyond the ECC of QR code (4-H). The following alalyis is similar to Section 4.1.

Let the two shares be: $\mathbf{A}_1,\ \mathbf{A}_2 \in \{\mathbf{S}_1,\ \mathbf{S}_2,\ \mathbf{S}_3\}$, where $\mathbf{A}_1$ is different from $\mathbf{A}_2$. To recover the secret QR code, the decoder stacks the two shares module by module:

$$\mathbf{H}_2(x,\ y) = \sqcap_{\ell=1}^{2} \mathbf{A}_\ell(x,\ y) \tag{32}$$

After this operation, $\mathbf{H}_2(x, y) \in \{\mathsf{R}, \mathsf{B}, \mathsf{C}, \mathsf{M}, \mathsf{Y}\}$. The blue modules are changed into cyan and yellow modules. Recall from $\mathcal{E}_0 = \{\mathsf{R}, \mathsf{B}\}$, $\mathcal{E}_1 = \{\mathsf{R}, \mathsf{G}, \mathsf{B}\}$, Equations (6), (7), (8) and (9), (10), (11), given a cyan or yellow module in $\mathbf{H}_2$, we have:

$$\Pr(\mathbf{H}(x, y) = 0 | \mathbf{H}_2(x, y) = \mathsf{C}) = 0, \tag{33}$$
$$\Pr(\mathbf{H}(x, y) = 1 | \mathbf{H}_2(x, y) = \mathsf{C}) = 1, \tag{34}$$
$$\Pr(\mathbf{H}(x, y) = 0 | \mathbf{H}_2(x, y) = \mathsf{Y}) = 0, \tag{35}$$
$$\Pr(\mathbf{H}(x, y) = 1 | \mathbf{H}_2(x, y) = \mathsf{Y}) = 1. \tag{36}$$

Given a red or blue module, we have:

$$\Pr(\mathbf{H}(x, y) = 0 | \mathbf{H}_2(x, y) = \mathsf{R}) = 1, \tag{37}$$
$$\Pr(\mathbf{H}(x, y) = 1 | \mathbf{H}_2(x, y) = \mathsf{R}) = 0, \tag{38}$$
$$\Pr(\mathbf{H}(x, y) = 0 | \mathbf{H}_2(x, y) = \mathsf{B}) = 1, \tag{39}$$
$$\Pr(\mathbf{H}(x, y) = 1 | \mathbf{H}_2(x, y) = \mathsf{B}) = 0. \tag{40}$$

a posteriori

To calculate other a posteriori probabilities, we need $\Pr(\mathbf{H}(x, y) = t_3)$, where $t_3 \in \{0, 1\}$, and $\Pr(\mathbf{H}_2(x, y) = t_4)$, where $t_4 \in \{\mathsf{R}, \mathsf{B}, \mathsf{C}, \mathsf{M}, \mathsf{Y}\}$. For the secret QR code, it is usually assumed that $\Pr(\mathbf{H}(x, y) = 0) = \Pr(\mathbf{H}(x, y) = 1) = 1/2$, because the masking operation balances the number of W/B modules. For the image $\mathbf{H}_2$, we can obtain:

$$\Pr(\mathbf{H}_2(x, y) = \mathsf{R}) = \frac{1}{2} \cdot \frac{1}{4} = \frac{1}{8}, \tag{41}$$
$$\Pr(\mathbf{H}_2(x, y) = \mathsf{B}) = \frac{1}{2} \cdot \frac{1}{2} = \frac{1}{8}, \tag{42}$$
$$\Pr(\mathbf{H}_2(x, y) = \mathsf{C}) = \frac{1}{2} \cdot \frac{1}{3} = \frac{1}{6}, \tag{43}$$
$$\Pr(\mathbf{H}_2(x, y) = \mathsf{M}) = \frac{1}{2} \cdot \frac{1}{2} + \frac{1}{2} \cdot \frac{1}{3} = \frac{5}{12}, \tag{44}$$
$$\Pr(\mathbf{H}_2(x, y) = \mathsf{Y}) = \frac{1}{2} \cdot \frac{1}{3} = \frac{1}{6}. \tag{45}$$

The a posteriori probability then can be determined from the Bayes rule:

$$\Pr(\mathbf{H}(x, y) = 0 | \mathbf{H}_2(x, y) = \mathsf{M}) = \frac{\frac{1}{2} \times \frac{1}{2}}{\frac{1}{2} \times \frac{1}{2} + \frac{1}{2} \times \frac{1}{3}} = \frac{3}{5}. \tag{46}$$

Similarly, we can get $\Pr(\mathbf{H}(x, y) = 1 | \mathbf{H}_2(x, y) = \mathsf{M}) = 2/5$. The above results are summarized in Table 3.

Table 3. Color probability diagram of two shares.

Color(c)	$\Pr(\mathbf{H}_2(x, y) = c)$	$\Pr(\mathbf{H}(x, y) = 0 \| \mathbf{H}_2(x, y) = c)$	$\Pr(\mathbf{H}(x, y) = 1 \| \mathbf{H}_2(x, y) = c)$
	$\frac{1}{8}$	1	0
	$\frac{1}{8}$	1	0
	$\frac{1}{6}$	0	1
	$\frac{5}{12}$	$\frac{3}{5}$	$\frac{2}{5}$
	$\frac{1}{6}$	0	1

Based on the above a posteriori probabilities, the MAP module detector decodes

$$\hat{\mathbf{H}}(x,\ y) = \begin{cases} 0, & \text{if } \mathbf{H}_2(x,\ y) \in \{\mathsf{R},\ \mathsf{B},\ \mathsf{M}\}, \\ 1, & \text{if } \mathbf{H}_2(x,\ y) \in \{\mathsf{C},\ \mathsf{Y}\}. \end{cases} \tag{47}$$

The probability of module recovery error can be calculated as:

$$\begin{aligned} \Pr(e_m) &= \Pr\{\hat{\mathbf{H}}(x,\ y) \neq \mathbf{H}(x,\ y)\} \\ &= \frac{5}{12}\Pr\left(\hat{\mathbf{H}}(x,\ y) = 0 | \mathbf{H}(x,\ y) = 1\right) \\ &= \frac{1}{6}. \end{aligned} \tag{48}$$

Since each codeword consists of 8 modules, the probability of codeword error e_c is:

$$\Pr(e_c) = 1 - \prod_{t=1}^{6}[1 - \Pr(e_m(t))] = 1 - \left(\frac{5}{6}\right)^8 = 0.767. \tag{49}$$

Similar to Equation (31), the probability of correct decoding should be:

$$\prod_{\ell=1}^{4}\sum_{q=1}^{8}\binom{25}{q}\Pr(e_c)^q\,[1 - \Pr(e_c)]^{25-q} = 4.73 \times 10^{-23}. \tag{50}$$

This means that the attacker needs to try on 10^{23} shares to get a correct decoding, which is almost impossible for ordinary computers nowadays.

In summary, given any two shares, it is also computational infeasible to decode the secret QR code correctly.

5. Experiment

This section presents our experimental results, including a $(3,\ 3)$-threshold VSS-QR experiment, the actual machine time of our VSS-QR (Section 5.1), experiments with different colors **T** (Section 5.2), experiments demonstrated in Section 4 (Section 5.3), and experiments of perfect module reconstruction (Section 5.4). For the actual machine time experiments (Section 5.1), we adopt QR code ("all versions"-H). All remaining experiments in this section adopt QR code (4-H). The conditions of computational experiment are shown in Table 4.

Table 4. The conditions of a computational experiment.

Hardware	Development Environment
CPU: Intel(R) Core(TM): i5-8500@ 3.00GHZ **Memory**: 8 G	MATLAB R2016b

5.1. Experiment Results

We assume that the encoder choose $\mathcal{E}_0 = \{\mathsf{R},\ \mathsf{B}\}$ and $\mathcal{E}_1 = \{\mathsf{R},\ \mathsf{G},\ \mathsf{B}\}$ as in Section 3.1. Secret QR code (Figure 4a) is divided into three meaningless images (Figure 4b–d). **T** (Figure 4e) can be recovered once the three shares are stacked. This **T** is not ordinary black and white QR code. It needs pre-processing to obtain black and white QR code $\mathbf{T}'$ (Figure 4f).

As shown in Figure 4g (White modules represent the difference between $\mathbf{T}'$ and **H**.), VSS-QR can completely reconstruct the secret QR code **H**.

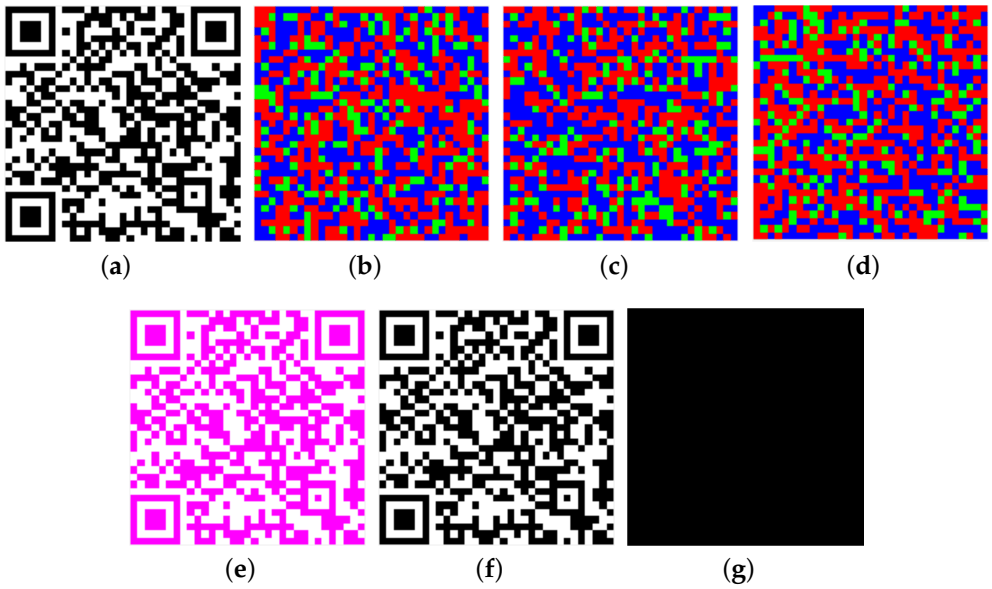

Figure 4. (**a**): **H**; (**b**): $\mathbf{S}_1$; (**c**): $\mathbf{S}_2$; (**d**): $\mathbf{S}_3$; (**e**): **T**; (**f**): $\mathbf{T}'$; (**g**): The difference images between **H** and $\mathbf{T}'$.

Section 3.3 gives the computational complexity VSS-QR algorithm and Fang's algorithm. This can be verified by the actual machine time. The commonly used versions of QR code in our daily life are mostly 4 and 5. When a module size is 41×41 pixels $(L = 41)$ and the printing resolution 600 DPI (Dots Per Inch), the size of them printed is around 6 centimeters (cm), which is the width of a ordinary phone screen. Therefore, this size of the module is used in the experiment. As shown in Figure 5, the time difference between them is very small when the version is very small. As the version gradually increases, the time difference is obviously different. Fang's algorithm is pixel-based and VSS-QR is module-based, so VSS-QR runs much faster when the version is larger. This result shows that the computational complexity of VSS-QR is significantly lower than Fang's algorithm.

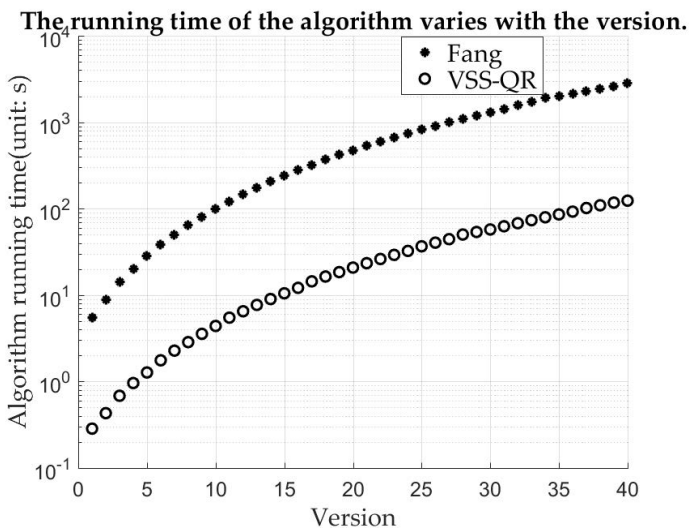

Figure 5. The running time of the algorithm varies with version.

5.2. Other Colors

This section shows experiments for other choices of $\mathcal{E}_0$ and $\mathcal{E}_1$, i.e., $\mathcal{E}_0 = \{\mathsf{G}, \mathsf{B}\}$, $\mathcal{E}_0 = \{\mathsf{R}, \mathsf{G}\}$, $\mathcal{E}_1 = \{\mathsf{R}, \mathsf{G}, \mathsf{B}\}$.

When $\mathcal{E}_0 = \{\mathsf{G},\ \mathsf{B}\}$, the color of **T** is cyan (Figure 6a). When $\mathcal{E}_0 = \{\mathsf{R},\ \mathsf{G}\}$, the color of **T** is yellow (Figure 6b). After pre-processing, they all become standard QR code (Figure 6c,d). As shown in Figure 6e,f (White modules represent the difference between them and **H**.), these standard QR code are the same as **H**. All secondary colors can be used for VSS-QR.

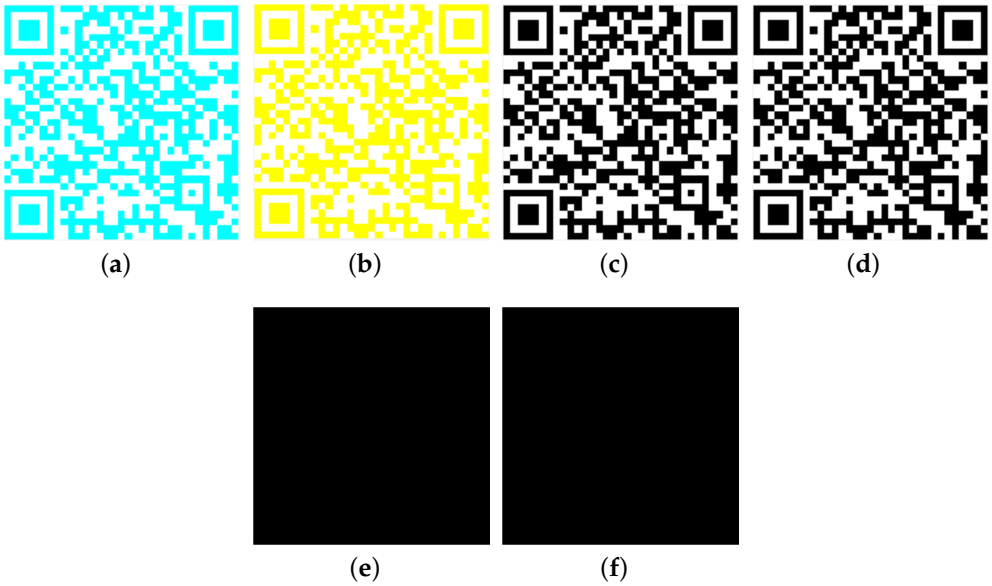

Figure 6. (**a**): $\mathbf{T}_c$; (**b**): $\mathbf{T}_y$; (**c**): $\mathbf{T'}_c$; (**d**): $\mathbf{T'}_y$; (**e**): The difference image between **H** and $\mathbf{T'}_c$; (**f**): The difference image between **H** and $\mathbf{T'}_y$.

5.3. Experiments for Security

Section 4 shows that when only one share or two shares have been obtained, the reconstructed "secret QR code" will not be decoded correctly by the standard decoder. This section verifies this conclusion experimentally.

Experiments test one and two shares respectively. As shown in Tables 5 and 6, the average of 10,000 experiments are recorded at a time. For QR code (4-H), there are four blocks. For each block, the Reed-Solomon code is $(\hat{c},\ \hat{k},\ \hat{r}) = (25,\ 9,\ 8)$ [23], where $\hat{c}$ is the total number of codewords in a block, $\hat{k}$ the number of data codewords, and $\hat{r}$ is the ECC. When the number of codeword errors per block exceeds 8, QR code will not be decoded by the standard decoder. From Table 5, we can see that the number of codeword errors in each block is about 24 ($24 > 8$). So, the reconstructed "secret QR code" is not decoded correctly by only using a share. From Table 6, we can see that the number of codeword errors in each block is about 19 ($19 > 8$). So, the reconstructed "secret QR code" is not decoded correctly by using any two shares. According to the experimental data, VSS-QR is secure.

Table 5. The result of experiment about reconstruct "secret QR code" by using one share.

Series Number	$\hat{\mathbf{n}}(\mathbf{e}_\mathbf{m}^1)$	$\hat{\mathbf{n}}(\mathbf{e}_\mathbf{m}^2)$	$\hat{\mathbf{n}}(\mathbf{e}_\mathbf{m}^3)$	$\hat{\mathbf{n}}(\mathbf{e}_\mathbf{m}^4)$
1	24.291	24.4871	24.2361	24.5258
2	24.2984	24.4949	24.2484	24.5226
3	24.2856	24.4965	24.2371	24.5253
4	24.3002	24.4888	24.2372	24.5141

Table 5. *Cont.*

Series Number	$\hat{\mathbf{n}}(\mathbf{e}_{\mathbf{m}}^{1})$	$\hat{\mathbf{n}}(\mathbf{e}_{\mathbf{m}}^{2})$	$\hat{\mathbf{n}}(\mathbf{e}_{\mathbf{m}}^{3})$	$\hat{\mathbf{n}}(\mathbf{e}_{\mathbf{m}}^{4})$
5	24.297	24.486	24.2269	24.5322
6	24.3092	24.4925	24.2502	24.5278
7	24.2879	24.4938	24.2259	24.5231
8	24.3024	24.4942	24.2352	24.5273
9	24.2877	24.4962	24.2464	24.5285
10	24.2905	24.5031	24.2419	24.5299

$\hat{\mathbf{n}}(\mathbf{e}_{\mathbf{m}}^{\hat{\mathbf{t}}})$ denotes the number of codeword errors in the $\hat{\mathbf{t}}$-th block, for $\hat{\mathbf{t}} = 1, 2, 3, 4$.

Table 6. The result of experiment about reconstruct "secret QR code" by using any two shares.

Series Number	$\hat{\mathbf{n}}(\mathbf{e}_{\mathbf{m}}^{1})$	$\hat{\mathbf{n}}(\mathbf{e}_{\mathbf{m}}^{2})$	$\hat{\mathbf{n}}(\mathbf{e}_{\mathbf{m}}^{3})$	$\hat{\mathbf{n}}(\mathbf{e}_{\mathbf{m}}^{4})$
1	19.0803	19.9352	19.4074	19.9332
2	19.048	19.9127	19.4067	19.9441
3	19.0621	19.9514	19.4229	19.9422
4	19.0651	19.9321	19.3731	19.9392
5	19.0079	19.9016	19.3957	19.9312
6	19.0141	19.9323	19.4219	19.9458
7	19.0258	19.9303	19.3769	19.903
8	19.0909	19.9517	19.4297	19.9377
9	19.0548	19.9258	19.3743	19.9386
10	19.0421	19.9196	19.4264	19.9417

$\hat{\mathbf{n}}(\mathbf{e}_{\mathbf{m}}^{\hat{\mathbf{t}}})$ denotes the number of codeword errors in the $\hat{\mathbf{t}}$-th block, for $\hat{\mathbf{t}} = 1, 2, 3, 4$.

5.4. Comparison with QRCSS

Similar to VSS-QR, QRCSS also divides the secret QR code into three shares. Every share is a standard QR code and six codewords are modified in each block. Because QR code has ECC capability, every share can be decoded correctly by standard QR code. In order to reconstruct secret QR code, three shares need to do "XOR". The reconstructed QR code $\mathbf{H}^r$ has some errors, but is lower than the ECC capability.

The experiments randomly added the same number of codeword errors to each block in $\mathbf{H}'$ and $\mathbf{H}^r$. Section 5.3 shows that the QR code (4-H) cannot be decoded by the standard decoder when the number of codeword errors per block exceeds 8. So, the number of codeword errors is from 1 to 9. When the number of codeword errors is 9, all of them cannot be decoded. Each experiment was performed 10,000 times. All results are shown in Table 7.

As the number of added-codeword-error increases, some $\mathbf{H}^r$ cannot be decoded by the standard decoder. When this number exceeds 4, $\mathbf{H}^r$ is difficult to decoded correctly by the standard decoder. Theoretically, when this number is less than 9, a few $\mathbf{H}^r$ can be decoded. All $\mathbf{H}^r$ cannot be decoded correctly when this number is greater than 4. However all $\mathbf{T}'$ that can be decoded by the standard decoder as this number increases. Until this number exceeds 8, $\mathbf{T}'$ cannot be decoded correctly. Compared with QRCSS, the biggest advantage of VSS-QR is that it can completely reconstruct the secret QR code, so VSS-QR preserves the ECC capability of QR code.

Table 7. ECC Test Experiment.

image	$\hat{n}(1)$	$\hat{n}(2)$	$\hat{n}(3)$	$\hat{n}(4)$	$\hat{n}(5)$	$\hat{n}(6)$	$\hat{n}(7)$	$\hat{n}(8)$	$\hat{n}(9)$
$\mathbf{H}^r$	10000	4975	1084	273	128	0	0	0	0
$\mathbf{T}'$	10000	10000	10000	10000	10000	10000	10000	10000	0

$\hat{n}(z)$ denotes the number of QR code that can be decoded by standard decoder when z codewords are changed randomly in each block.

5.5. Summary of Experiments

As shown in Table 8, VSS-QR has two advantages. Comparing with [24], the run time of VSS-QR is faster than that of it. Comparing with [26], VSS-QR does not sacrifice ECC capability of QR code. VSS-QR inherits the merits of both of them.

Table 8. Analysis of different algorithms.

	Secret Image is Completely Restored	Run Time	Restored QR Code Can Be Decoded
[24]	Yes	Slow	Yes
[26]	No	Fast	Yes
VSS-QR	Yes	Fast	Yes

VSS-QR is module-based, so the main factor affecting the running speed of it is the version of QR code. The larger the version, the slower it will run.

6. Extension to (k, n)-threshold VSS-QR

The proposed $(3,\ 3)$-threshold VSS-QR can be extended to $(k,\ n)$-threshold VSS-QR. This section presents a design of $(k,\ n)$-threshold VSS-QR.

To design $(k,\ n)$-threshold VSS-QR, the value of n and k need to be determined. The range of n is $3 \leq n \leq max(\hat{n}_0,\ \hat{n}_1)$, where $\hat{n}_0$ is the number of black modules and $\hat{n}_1$ is the number of white modules in $\mathbf{H}$. In order to determine k, we need determine the following four parameters:

1. l: the number of modules that can be modified in every share.
2. $\hat{r}$: the ECC capability of QR code in every block.
3. $\hat{n}_b$: the number of block in QR code.
4. $\Pr(e_c)$ from Equation (49).

The number of changeable modules l can be calculated as

$$l = \left\lceil \frac{max(\hat{n}_0,\ \hat{n}_1)}{n-2} \right\rceil. \tag{51}$$

Assuming that 8 modules in a codeword are all wrong, we can determine k_1 as

$$\begin{aligned} \frac{\Pr(e_c) \cdot D^2 - k_1 \cdot l}{D^2} &= \frac{8\hat{r} \cdot \hat{n}_b}{D^2} \\ 0.767D^2 - k_1 \cdot l &= 8\hat{r} \cdot \hat{n}_b \\ k_1 &= \left\lceil \frac{0.767D^2 - 8\hat{r} \cdot \hat{n}_b}{l} \right\rceil. \end{aligned} \tag{52}$$

When $k \geq k_1$, the reconstructed QR code may be decoded by using k shares. However, this QR code has some errors. If we want to completely reconstruct the secret QR code, all shares must be obtained.

Encoding algorithm for $\mathbf{S}_1$ and $\mathbf{S}_2$ are the same as proposed in Section 3.1. Therefore, $\mathbf{S}_1$ and $\mathbf{S}_2$ must be obtained when decoding. In the following shares, encoding rule is changed. Only l modules are modified in next every share. The encoding algorithm is shown in Algorithm 2. Decoding method is the same as proposed in Section 3.2.

Algorithm 2 (k, n)-threshold VSS-QR algorithm

Input:
Secret QR code $\mathbf{H}$
Output:
n shares: $\mathbf{S}_1,\ \mathbf{S}_2,\ \ldots,\ \mathbf{S}_n$.

Determine the two sets $\mathcal{E}_0$ and $\mathcal{E}_1$:
$\mathcal{E}_1 = \{\mathsf{R},\ \mathsf{G},\ \mathsf{B}\}, \mathcal{E}_0 \sim \{\{\mathsf{G},\ \mathsf{B}\},\ \{\mathsf{R},\ \mathsf{B}\}, \{\mathsf{R},\ \mathsf{G}\}\}$.

for x from 1 to D **do**
  for y from 1 to D **do**
    if $\mathbf{H}(x,\ y) = 0$ **then**
      $\mathbf{S}_1(x\ ,y) \sim \mathcal{E}_0$,
      $\mathbf{S}_2(x,\ y) \sim \mathcal{E}_0$,
      $\mathbf{S}_3(x,\ y) \leftarrow \mathbf{S}_1(x,\ y)$,
      ...
      $\mathbf{S}_n(x,\ y) \leftarrow \mathbf{S}_1(x,\ y)$.
    else
      $\mathbf{S}_1(x,\ y) \sim \mathcal{E}_1$,
      $\mathbf{S}_2(x,\ y) \sim \mathcal{E}_1 \backslash \{\mathbf{S}_1(x,\ y)\}$,
      $\mathbf{S}_3(x,\ y) \leftarrow \mathbf{S}_1(x,\ y)$,
      ...
      $\mathbf{S}_n(x,\ y) \leftarrow \mathbf{S}_1(x,\ y)$.
    end if
  end for
end for

for $\hat{i}$ from 3 to n **do**
  for $\hat{j}$ from 1 to l **do**
    Randomly select position $(x,\ y)$, where $\mathbf{H}(x,\ y) = 0$ that $1 \leq x,\ y \leq D$. Make sure that $(x,\ y)$ can only be repeated when all $(x,\ y)$ appear once. Randomly select u, where $3 \leq u \leq n$ that the number from 3 to n can only appear once.
    $\mathbf{S}_u(x,\ y) \sim \mathcal{E}_0 \backslash \{\mathbf{S}_1(x,\ y)\}$.
    Randomly select position $(x,\ y)$, where $\mathbf{H}(x,\ y) = 1$ that $1 \leq x,\ y \leq D$. Make sure that $(x,\ y)$ can only be repeated when all $(x,\ y)$ appear once. Randomly select v, where $3 \leq v \leq n$ that the number from 3 to n can only appear once.
    $\mathbf{S}_v(x,\ y) \sim \mathcal{E}_1 \backslash \{\mathbf{S}_1(x,\ y), \mathbf{S}_2(x,\ y)\}$.
  end for
end for

for x from 1 to D **do**
  for y from 1 to D **do**
    $\mathbf{S}_\ell[i,\ j] = \mathbf{H}(x,\ y)$, for $i \in [(x-1)L+1,\ xL]$, $j \in [(y-1)L+1,\ yL]$, $\ell = 1,\ 2,\ \ldots,\ n$.
  end for
end for

7. Conclusions

In this paper, we propose a $(3, 3)$ color visual secret sharing for QR Code with perfect module reconstruction. The new VSS-QR is module-based, which overcomes the low encoding speed disadvantage of pixel-based algorithm. It also ensures that the secret QR code can be completely reconstructed. The experiment results show that it is secure under the condition that the computational device available to the attacker is limited to standard QR code decoder.

The security of VSS-QR does not belong to perfect security but belongs to the computational security. It is secure under the assumption that the attacker's tool is limited to QR code decoder. The method we proposed is to use multiple colors that can also be extended. In our future work, a method will be designed to improve the security of QR code, and VSS will utilize multiple tones instead of multiple colors.

Author Contributions: Conceptualization, T.L. and B.Y.; software, T.L., B.Y.; Formal analysis, T.L., B.Y. and J.-S.P.; Methodology, T.L., B.Y. and J.-S.P.; Writing—original draft, T.L. and B.Y.; Writing—review & editing, T.L., B.Y. and J.-S.P.

Funding: This research was funded by NATURAL SCIENCE FOUNDATION OF CHINA 61571139 and 61272432, SHANDONG PROVINCIAL NATURAL SCIENCE FOUNDATION ZR2014JL044.

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

References

1. Chen, C.M.; Xiang, B.; Liu, Y.; Wang, K.H. A secure authentication protocol for Internet of vehicles. *IEEE Access* **2019**, *7*, 12047–12057. [CrossRef]
2. Chen, C.M.; Wang, K.H.; Yeh, K.H.; Xiang, B.; Wu, T.Y. Attacks and solutions on a three-party password-based authenticated key exchange protocol for wireless communications. *J. Ambient. Intell. Humaniz. Comput.* **2019**, *10*, 3133–3142. [CrossRef]
3. Pan, J.S.; Kong, L.; Sung, T.W.; Tsai, P.W.; Snášel, V. α-Fraction first strategy for hierarchical model in wireless sensor networks. *J. Internet Technol.* **2018**, *19*, 1717–1726.
4. Wu, T.Y.; Chen, C.M.; Wang, K.H.; Meng, C.; Wang, E.K. A provably secure certificateless public key encryption with keyword search. *J. Chin. Inst. Eng.* **2019**, *42*, 20–28. [CrossRef]
5. Pan, J.S.; Lee, C.Y.; Sghaier, A.; Zeghid, M.; Xie, J. Novel Systolization of Subquadratic Space Complexity Multipliers Based on Toeplitz Matrix-Vector Product Approach. *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.* **2019**, *27*, 1614–1622. [CrossRef]
6. Lin, P.Y.; Chen, Y.H. High payload secret hiding technology for QR codes. *EURASIP J. Image Video Process.* **2017**, *2017*, 14. [CrossRef]
7. Huang, P.C.; Chang, C.C.; Li, Y.H.; Liu, Y. High-payload secret hiding mechanism for QR codes. *Multimed. Tools Appl.* **2019**, *78*, 22331–22350. [CrossRef]
8. Huang, H.C.; Chang, F.C.; Fang, W.C. Reversible data hiding with histogram-based difference expansion for QR code applications. *IEEE Trans. Consum. Electron.* **2011**, *57*, 779–787. [CrossRef]
9. Thodi, D.M.; Rodriguez, J.J. Expansion Embedding Techniques for Reversible Watermarking. *IEEE Trans. Image Process.* **2007**, *16*, 721–730. [CrossRef]
10. Shi, Y.; Li, X.; Zhang, X.; Wu, H.; Ma, B. Reversible data hiding: Advances in the past two decades. *IEEE Access* **2016**, *4*, 3210–3237. [CrossRef]
11. Weng, S.; Pan, J.S. Reversible watermarking based on two embedding Schemes. *Multimed. Tools Appl.* **2016**, *75*, 7129–7157. [CrossRef]
12. Weng, S.; Zhang, G.; Pan, J.S.; Zhou, Z. Optimal PPVO-based reversible data hiding. *J. Vis. Commun. Image Represent.* **2017**, *48*, 317–328. [CrossRef]
13. Weng, S.; Pan, J.S.; Zhou, L. Reversible data hiding based on the local smoothness estimator and optional embedding strategy in four prediction modes. *Multimed. Tools Appl.* **2017**, *76*, 13173–13195. [CrossRef]
14. Hong, W.; Zhou, X.; Weng, S. Joint adaptive coding and reversible data hiding for AMBTC compressed images. *Symmetry* **2018**, *10*, 254. [CrossRef]
15. Weng, S.; Chen, Y.; Ou, B.; Chang, C.C.; Zhang, C. Improved K-Pass Pixel Value Ordering Based Data Hiding. *IEEE Access* **2019**, *7*, 34570–34582. [CrossRef]
16. Weng, S.; Zhao, Y.; Pan, J.S.; Ni, R. A novel reversible watermarking based on an integer transform. In Proceedings of the 2007 IEEE International Conference on Image Processing, San Antonio, TX, USA, 16 September–19 October 2007; Volume 3, pp. 241–244.
17. Naor, M.; Shamir, A. *Visual Cryptography*; Springer: Boston, MA, USA, 1994; pp. 1–12, ISBN 978-3-540-60176-0.
18. Hou, Y.C. Visual cryptography for color images. *Pattern Recognit.* **2003**, *36*, 1619–1629. [CrossRef]

19. Fang, W.P.; Lin, J.C. Visual cryptography with extra ability of hiding confidential data. *J. Electron. Imaging* **2006**, *15*, 023020. [CrossRef]
20. Lin, S.J.; Lin, J.C.; Fang, W.P. Visual Cryptography (VC) with non-expanded shadow images: Hilbert-curve approach. In Proceedings of the 2008 IEEE International Conference on Intelligence and Security Informatics, Taipei, Taiwan, 17 June 2008; pp. 271–272.
21. Fang, W.P. Friendly progressive visual secret sharing. *Pattern Recognit.* **2008**, *41*, 1410–1414. [CrossRef]
22. Suklabaidya, A.; Sahoo, G. Visual cryptographic applications. *Int. J. Comput. Sci. Eng.* **2013**, *5*, 464.
23. ISO, B. IEC 16022: Information Technology-Automatic Identification and Data Capture Techniques-Data Matrix Bar Code Symbology Specification. *BS ISO/IEC* **2006**, *16022*.
24. Fang, W.P. Offline QR code authorization based on visual cryptography. In Proceedings of the 2011 Seventh International Conference on Intelligent Information Hiding and Multimedia Signal Processing, Dalian, China, 14–16 October 2011; pp. 89–92.
25. Lin, P.Y. Distributed secret sharing approach with cheater prevention based on QR code. *IEEE Trans. Ind. Inform.* **2016**, *12*, 384–392. [CrossRef]
26. Chow, Y.W.; Susilo, W.; Yang, G.; Phillips, J.G.; Pranata, I.; Barmawi, A.M. Exploiting the error correction mechanism in QR codes for secret sharing. In *Australasian Conference on Information Security and Privacy*; Springer: Cham, Switzerland, 2016, pp. 409–425.
27. Cheng, Y.; Fu, Z.; Yu, B. Improved Visual Secret Sharing Scheme for QR Code Applications. *IEEE Trans. Inf. Forensics Secur.* **2018**, *13*, 2393–2403. [CrossRef]
28. Shyu, S.J. Image Encryption by Random Grids. *Pattern Recognit.* **2007**, *40*, 1014–1031. [CrossRef]
29. Gonzalez, R.C.; Wintz, P. *Digital Image Processing*; Addison-Wesley Publishing Co., Inc. Applied Mathematics and Computation: Reading, MA, USA, 1977; pp. 290–294.
30. Cimato, S.; Yang, C.N. *Visual Cryptography and Secret Image Sharing*; CRC Press: Boca Raton, FL, USA, 2017; pp. 38–39.
31. Burger, W.; Burge, M.J. *Principles of Digital Image Processing: Fundamental Techniques*; Springer: Boston, MA, USA, 2010; pp. 200–204, ISBN 978-1-84800-190-9.
32. Otsu, N. A threshold selection method from gray-level histograms. *IEEE Trans. Syst. Man Cybern.* **1979**, *9*, 62–66. [CrossRef]
33. Zxing Library. 2019. Available online: https://github.com/zxing/zxing (accessed on 30 September 2015).
34. Kay, S. *Fundamentals of Statistical Signal Processing: Detection Theory*; Prentice Hall: Upper Saddle River, NJ, USA, 1998.