

Article

JPEG-Resistant Robust Watermarking with Applications to Visual Secret Sharing Techniques

Hsiang-Cheh Huang ¹, Jia-En Li ¹ and Feng-Cheng Chang ^{2,*}

¹ Department of Electrical Engineering, National University of Kaohsiung, Kaohsiung City 811726, Taiwan; hchuang@nuk.edu.tw (H.-C.H.); lije901@gmail.com (J.-E.L.)

² Department of Computer Science and Information Engineering, Tamkang University, New Taipei City 251301, Taiwan

* Correspondence: 135170@o365.tku.edu.tw

Abstract

Robust watermarking has emerged as a pivotal field in digital security, particularly for protecting intellectual property. This study introduces a robust watermarking framework specifically engineered to withstand JPEG compression attacks. To evaluate our scheme, several performance metrics should be considered, including robustness against intentional JPEG compression, embedding capacity, and the perceptual quality of watermarked images. For practical applications, we choose the quick response code to serve as the watermark, and we extend our evaluation to high-resolution pictures, both captured by the authors and obtained on the Internet with no copyright issues. These images possess resolutions that are dozens-of-times higher than conventional test images, facilitating a vastly increased capacity. This expanded capacity makes the integration of visual secret sharing practical, adding an additional layer of cryptographic protection to the watermarking scheme. Simulation results present the effectiveness of our implementation, demonstrating that our approach achieves a superior balance between data integrity and visual fidelity, even when subjected to rigorous attack conditions.

Keywords: robust watermarking; JPEG; discrete cosine transform; quick response code; visual secret sharing; imperceptibility; capacity

1. Introduction

Robust watermarking plays an important role in information security research. It has emerged as a technical feature and a necessity for copyright protection and authenticity [1,2]. Robust watermarking embeds the watermark directly into the pixel values or frequency coefficients of the media. This ensures that the identifying information is mathematically bound to the content itself that survives the challenges of distribution [3,4]. We focus on natural images, and applying watermarking to hyperspectral images is indeed a fascinating and critical area, especially for protecting sensitive remote sensing or medical data [5]. Watermarks in binary form can be embedded into the original image with algorithms devised by researchers to obtain the watermarked image [6–8]. Quick response codes (QR codes), which are commonly seen nowadays, are suitable to play the role of the watermark [9,10]. After completing the watermark-embedding procedures, we have the watermarked image, and its quality should be sufficient to make the embedded watermark imperceptible. Watermarked images may experience intentional signal processing,

Academic Editor: Priyadarsi Nanda

Received: 25 June 2026

Revised: 16 July 2026

Accepted: 17 July 2026

Published: 26 July 2026

Copyright: © 2026 by the authors.

Licensee MDPI, Basel, Switzerland.

This article is an open access article distributed under the terms and conditions of the [Creative Commons Attribution \(CC BY\)](https://creativecommons.org/licenses/by/4.0/) license.

called attacks, during transmission [11]. We expect to extract the watermark bits from the attacked image. When the error rate between extracted and embedded watermarks is below a certain threshold, we may call it a robust watermarking algorithm.

For assessing the performances of algorithms, for fair comparisons, researchers often chose test images from major databases [12]. Watermarked image quality, called imperceptibility, and the capability of withstanding attacks, called robustness, are often compared with the embedding of the same amount of watermark bits. Resolutions of conventional test images generally range from 256×256 to 1024×1024 , implying 0.25 to 1.0 megapixels (MP). Compared to smartphone cameras in 2026, which generally have resolutions of 48 MP, conventional test images are too small to meet practical applications. In addition, there are also many images with high resolutions on the Internet [13]. Images with higher resolutions imply a greater number of bits to be embedded. Hence, we also apply visual secret sharing (VSS) to serve as the watermark [14,15] and perform watermark embedding and extraction with a devised scheme.

In this study, we present a unified framework integrating quick response codes and visual secret sharing with the high-resolution pictures, both captured by the authors and obtained on the Internet with no copyright issues. The primary contributions include better or comparable imperceptibility based on watermarking conventional test images, the robustness of JPEG attacks over a wide range of compression settings, and the vast amounts of bits that can be embedded into pictures captured by us. Comprehensive evaluations and discussions demonstrate the feasibility of our scheme and corresponding framework for practical applications.

2. Fundamentals with Robust Watermarking and Embedded Watermark

2.1. Elements for Robust Watermarking

For robust watermarking, at the encoder, the luminance values of the pixels or the transform coefficients relating to original image X should be altered for embedding the watermark W , and watermarked image X_w is obtained. The watermarked image is delivered to the decoder, with the expectation of experiencing possible attacks during transmission. Upon reception of attacked image X' at the decoder, only the watermark W' needs to be extracted from attacked image. The attacked image is then discarded because the original image need not be recovered with the arrangements in robust watermarking. The extracted watermark might be influenced by attacks during transmission, so the embedded watermark and the extracted one might not be identical, and this would be a major reason for devising robust watermarking schemes.

There are several performance parameters for robust watermarking, including the following aspects:

1. Imperceptibility;
2. Embedding capacity;
3. Robustness.

Imperceptibility, or watermarked image quality, refers to the difference between the original image, X , and the watermarked one, X_w , at the encoder. The watermark W is hidden in X with algorithms devised. The peak signal-to-noise ratio (PSNR), calculated objectively, and structural similarity (SSIM), evaluated from a subjective viewpoint, are commonly employed [16–18]. The PSNR can be directly calculated from the mean squared error (MSE) between the two, while the SSIM is built to mimic human visual perception by isolating structural shifts from lighting modifications.

We can calculate the mean squared error between X and X_w . Note that the MSE between the original image X and the watermarked one X_w , with resolutions of $M \times N$, can be calculated by

$$\text{MSE} = \frac{1}{M \cdot N} \sum_{i=0}^{M-1} \sum_{j=0}^{N-1} (X_w(i, j) - X(i, j))^2 \quad (1)$$

It is easily observed that $\text{MSE} \geq 0$. If the MSE equals 0, it implies they are identical. PSNR is an objective measure with the unit of decibel (dB), which directly corresponds to MSE. Next, PSNR can be calculated with

$$\text{PSNR} = 10 \cdot \log_{10} \left(\frac{255^2}{\text{MSE}} \right) \text{ (dB)}. \quad (2)$$

In real-world applications, for PSNR larger than 38 dB, it could be classified as good or excellent. SSIM relates to a subjective rating of image quality [18], which may work together with PSNR for performance assessments. Practically, for SSIM larger than 0.9, it could be classified as good or excellent.

Embedding capacity implies the bitstream length C , or the number of bits, of the watermark to be embedded at the encoder. Capacity would be expected to be as large as possible. However, embedding a larger capacity into the same image would directly cause more degradation, which would have opposite effects on imperceptibility. People generally choose random bitstreams to play the role of the watermark. It would be beneficial to choose watermarks that convey meaningful information, and QR codes would serve as a good candidate. To extend this concept, we may try integrating visual secret sharing into our implementation. A large value of C would introduce noticeable visual alterations to the original image, usually leading to a large MSE value. Hence, there might be a balance between imperceptibility and embedding capacity. It can be easily expected that for images with higher resolutions, more bits can be embedded under the similar MSE constraint. Hence, a popular metric related to the capacity is the capacity ratio, or ratio of bitstream length to image resolution, or $\frac{C}{M \times N}$, for fair comparisons between algorithms with different image sizes.

Robustness means the capability to withstand intentional attacks on a watermarked image at the decoder [3,4]. There are a variety of attacks, like JPEG recompression attacks, filtering attacks, geometric attacks, noise addition attacks, and so on. In this study, we focus on JPEG recompression attacks and present our simulations and performances. After experiencing attacks during transmission, watermarked image X_w becomes attacked image X' . Extracted watermark W' , which is extracted from X' , needs to compare to embedded one W . Robustness is presented with the bit correct rate (BCR), implying the percentage of correctly extracted bits in the extracted watermark:

$$\text{BCR} = \left[1 - \frac{1}{C} \sum_{k=1}^C (W(k) \oplus W'(k)) \right] \times 100\%. \quad (3)$$

In Equation (3), the exclusive-OR (XOR) operator, $\oplus$, is applied for the calculation. We can also use the exclusive-NOR (XNOR) operator to check the differences between embedded bit $W(k)$ and extracted one $W'(k)$ and calculate the BCR. We expect to have a larger BCR value. When BCR equals 100%, it means the embedded watermark and extracted one are identical. When we consider the robustness of the algorithm after attacks are experienced, we may choose a threshold for BCR to examine the level of robustness. In addition to choosing BCR, we can also calculate other measures such as normalized cross-correlation (NC) or scan an extracted QR code directly with our smartphone to examine whether the extracted QR code can be recognizable for the evaluation of robustness.

2.2. Watermarking with Quantization Index Modulation in Discrete Cosine Transform Domain

For robust watermarking, the luminance values of the pixels in the spatial domain, or the transform coefficients in the frequency domain relating to the original image,

should be altered for watermark embedding. Based on conventional research, embedding the watermark in the frequency domain, or transform domain, would result in better robustness after experiencing attacks. In this study, we apply discrete cosine transform (DCT) for watermark embedding and extraction. The DCT spectrum is utilized to determine specific coefficients that are then employed in the embedding process with binary watermarks [19–22]. The embedding mechanism itself is implemented using quantization index modulation (QIM) techniques [23,24].

Discrete cosine transform is widely used today. It plays one important role in the JPEG (Joint Photographic Experts Group) international standard. With DCT, the original image is subjected to partition into 8×8 non-overlapping blocks. Then, with the 64 luminance values, $X(i, j)$, $i, j = 0, 1, \dots, 7$, in one 8×8 block, the 64 DCT coefficients, $Y(u, v)$, $u, v = 0, 1, \dots, 7$, are calculated with Equation (4):

$$Y(u, v) = \frac{1}{4} \alpha(u) \alpha(v) \sum_{i=0}^7 \sum_{j=0}^7 X(i, j) \cos\left(\frac{(2i+1)u\pi}{16}\right) \cos\left(\frac{(2j+1)v\pi}{16}\right), \quad (4a)$$

$$\alpha(u) = \begin{cases} \frac{1}{\sqrt{2}}, & u = 0 \\ 1, & u = 1, 2, \dots, 7. \end{cases} \quad (4b)$$

Then, $Y(u, v)$ is placed with a zigzag scanning order, as depicted in Figure 1, to form the coefficient vector Y ,

$$Y = \{Y(0), Y(1), Y(2), \dots, Y(62), Y(63)\}. \quad (5)$$

Position with number 0 denotes the DC coefficient $Y(0)$ of such a block, while positions with numbers from 1 to 63 correspond to AC coefficients, $Y(1)$ to $Y(63)$, from low-frequency to high-frequency coefficients in 8×8 DCT (as shown in Figure 1).

0	1	5	6	14	15	27	28
2	4	7	13	16	26	29	42
3	8	12	17	25	30	41	43
9	11	18	24	31	40	44	53
10	19	23	32	39	45	52	54
20	22	33	38	46	51	55	60
21	34	37	47	50	56	59	61
35	36	48	49	57	58	62	63

Figure 1. One 8×8 block containing 64 DCT coefficients. Positions of coefficients follow the zigzag scanning order. Positions in blue are candidate coefficients for watermark embedding.

For watermark embedding, some of the DCT coefficients in every block should be chosen and altered. With DCT, it concentrates most of the energy into the upper-left corner of the 8×8 block. The DC coefficient is proportional to the average luminance value of the whole 8×8 block, and it often holds up to a very large portion of the energy of the block. Thus, alteration of the DC coefficient would result in large MSE and degradation in imperceptibility. With this in mind, embedding the watermark into AC coefficients would be a reasonable choice. For AC coefficients, low-frequency coefficients correspond to smooth gradients or flat surfaces, where a large portion of the total AC energy is packed into this range. For high-frequency coefficients, they relate to sharp edges or fine textures,

which drop to nearly zero due to the energy concentration of DCT. Therefore, embedding into both extremes would have drawbacks. With this observation, embedding into low-frequency coefficients would severely affect imperceptibility from both objective and subjective viewpoints, and embedding into high-frequency coefficients would have limited capability to withstand attacks. With the discussions above, embedding the watermark into middle-frequency coefficients would be the compromise to look for a balance between imperceptibility and robustness. Researchers have focused on the proper selection of coefficients with optimization techniques [25–28]. Employing optimization techniques for watermarking with properly designed fitness functions would be a rigorous scheme. However, training with optimization techniques would consume lots of time, and, hence, we refer to the results to serve as a rule of thumb for the selection of coefficient position for embedding. In this study, we choose the third, fourth, and fifth diagonal [16] in Figure 1 with positions in blue, ranging from AC3 to AC14 and selected with a pseudo-random number generator (PRNG) to perform watermark embedding with quantization index modulation. The PRNG seed functions as a symmetric cryptographic key shared (out-of-band) between the encoder and decoder. The proposed scheme is a blind watermarking method; the decoder does not require the original image, but it does require the identical PRNG seed to initialize its state. Because the PRNG is deterministic, initializing it with the shared seed guarantees that the decoder generates the exact same sequence of pseudo-random indices as the encoder. For example, if the seeded PRNG selects AC10 for a specific block at the encoder, the decoder's seeded PRNG will identically pinpoint AC10 for extraction.

In QIM, we apply two scalar quantizers corresponding to the embedding of bit 0 and bit 1 [23,24]. Suppose we have the input y , which corresponds to the selected AC coefficient in a certain block. We choose the quantization stepsize Δ and perform watermark embedding based on the bit values. For embedding bit 0 or bit 1, marked value $\hat{y}$ is calculated with Equation (6a) and Equation (6b), respectively.

$$\hat{y} = Q_0(y) = \Delta \cdot \text{round}\left(\frac{y}{\Delta}\right); \quad (6a)$$

$$\hat{y} = Q_1(y) = \Delta \cdot \left(\text{round}\left(\frac{y}{\Delta}\right) + 0.5\right). \quad (6b)$$

Here, $Q_0(\cdot)$ and $Q_1(\cdot)$ are two scalar quantizers. We can easily observe that after embedding bit 0, $\hat{y}$ has the value of multiples of Δ , or even multiples of $\frac{\Delta}{2}$. Also, after embedding bit 1, $\hat{y}$ has the value of odd multiples of $\frac{\Delta}{2}$. This observation facilitates the data extraction procedures. We can check the relationships between $\hat{y}$ and $\frac{\Delta}{2}$. If $\hat{y}$ is even multiples of $\frac{\Delta}{2}$, the extracted bit is represented by bit 0; otherwise, it is bit 1. When no attack applies, watermark extraction works perfectly. However, due to the possible attacks that may be experienced, there might be the chance of erroneously extracted watermark bit values after extraction, and we can check the BCR value for accessing the robustness. When we choose a smaller value of Δ , it introduces smaller errors to maintain the quality of the watermarked image, but it may reduce the capability to correctly extract the watermark bit. In contrast, when we choose a larger value of Δ , it would have good robustness with the sacrifice of watermarked image quality. In our simulations, we choose the range of Δ between 1.9 and 2.6 with different images. The rule of thumb is to look for a balance between watermarked image quality and the robustness for a large portion of JPEG attacks.

After obtaining the marked DCT coefficient with QIM, along with other DCT coefficients that are unchanged in the same block, we have the marked coefficient vector in the transform domain. Suppose that we plan to embed 1 bit in such a block, and AC10 is

selected by PRNG, for instance. Then, corresponding to Equations (4) and (6), the marked coefficient vector of such a block becomes

$$\hat{Y} = \{Y(0), Y(1), \dots, Y(9), \hat{Y}(10), Y(11), \dots, Y(62), Y(63)\}. \quad (7)$$

In Equation (7), only AC10 is altered, and the other 63 coefficients are kept intact. Next, we use the zigzag scanning order to place the coefficients in two-dimensional form in Figure 1 and perform inverse DCT to $\hat{Y}$ to acquire the marked block. Inverse DCT of one 8×8 block is calculated with Equation (8):

$$\hat{X}(i, j) = \frac{1}{4} \sum_{u=0}^7 \sum_{v=0}^7 \alpha(u) \alpha(v) \hat{Y}(u, v) \cos\left(\frac{(2i+1)u\pi}{16}\right) \cos\left(\frac{(2j+1)v\pi}{16}\right). \quad (8)$$

Here, $\alpha(u)$ and $\alpha(v)$ can be represented by Equation (4b). When all blocks complete this procedure, we can group them together and obtain the watermarked image X_w . The imperceptibility, represented by PSNR and SSIM, between X_w and X can be calculated. Next, the watermarked image X_w is expected to experience attacks during transmission, and we denote the attacked image as X' .

For the extraction of embedded bits from attacked image X' , we first calculate the DCT coefficients of X' block by block, choose the selected DCT coefficient in every block with PRNG, and extract the watermark bit of the block with QIM decoding by determining the odd or even multiples of $\frac{\Delta}{2}$ in the extracted value. BCR between extracted and embedded watermarks in Equation (3) needs to be calculated to present the robustness of withstanding attacks.

2.3. Embedded QR Code with Visual Secret Sharing Capability

For conventional watermarking, researchers tend to choose random bitstreams to serve as embedded watermarks. In addition, for fair comparisons of performances among different algorithms proposed in the literature, researchers select commonly employed test images, like the USC-SIPI Image Database [12], with image sizes between 256×256 and 1024×1024 , leading to the resolutions of 0.25 megapixel (MP) to 1 MP. The performance parameters mentioned in Section 2.1, including imperceptibility, capacity, and robustness, are compared based on the test images selected. From a practical point of view, at the time of submission of this study in June 2026, the average primary camera resolution for mainstream smartphones sits around 48 MP, and there are lots of high-resolution images on the Internet [13]. When we choose to embed 1 bit per block with DCT, we can hide $\frac{1024}{8} \times \frac{1024}{8} = 16,384$ bits for the 1 MP image and $\frac{8192}{8} \times \frac{6144}{8} = 786,432$ bits for the 48 MP image. With the vast increase in resolutions, we may hide more bits in the original image, which may be meaningful information rather than random bitstreams. In this study, we consider employing visual secret sharing, based on quick response codes (QR codes), to serve as the application for robust watermarking.

The definition of a QR code is precisely specified in ISO/IEC 18004:2024 [9]. QR codes are two-dimensional barcodes that store information, such as webpage links, telephone numbers, short message service, or plain texts, that can be scanned by smartphone cameras [10]. They can be seen in many places; for instance, consumers in a restaurant can order meals by scanning the provided QR code, linking to the designated webpages. QR codes are often seen as black-and-white squares, called modules. The number of modules in one QR code directly relates to its version. There are 40 versions of QR codes, and the size of a QR code is strictly determined by its version number. The standard formula for the side length of a QR code in modules is $(17 + 4 \cdot v)$, where v represents the version number ranging from 1 to 40. Each increase in the version number adds exactly four modules to both the width and the height. This incremental growth allows developers to choose the smallest possible matrix size that can still safely hold their required embedding capacity. All QR codes feature the three large, iconic finder patterns, or the squares in the

upper-left, upper-right, and lower-left corners. Each finder pattern is composed of 7×7 modules, which helps the smartphone camera or scanner to better locate the QR code and recognize the information therein. Figure 2 is a demonstration of the original image and the QR code. Figure 2a is a test image *male* with size 1024×1024 . In Figure 2b, for preparing the Uniform Resource Locator (URL) of <https://sipi.usc.edu/database> where Figure 2a is originated, we choose a version 6 QR code, meaning that there are 41×41 modules. Users can determine the module size, and, here, we set the module size to be 3×3 , leading to the result that the QR code has the resolutions of 123×123 . It results in 15129 bits for embedding, which is the maximum allowable capacity in a QR code that fits the embedding capacity of 16384 bits in the *male* image. Figures 2a and 2b are scaled proportionally according to their resolutions for comparisons.

Next, we apply the concept of visual secret sharing with the consideration of security issues [14,15]. With the premise that images captured with smartphone cameras can hide dozens of times more bits than those with conventional test images, we choose VSS for practical applications due to increased amounts of capacity and the capability in security. Information relating to QR codes can be calculated to produce several shares in VSS, which look like random patterns, for hiding in images captured with smartphone cameras. Marked images may experience attacks, and watermarks are extracted from attacked images, as described in Section 2.1. Once the number of shares reaches the pre-determined threshold, the mixture of shares can be calculated to obtain reconstructed QR codes. In addition to examining the BCR value for robustness, reconstructed QR codes obtained from VSS decoding can be validated with the scanning of smartphone cameras to check whether it works.

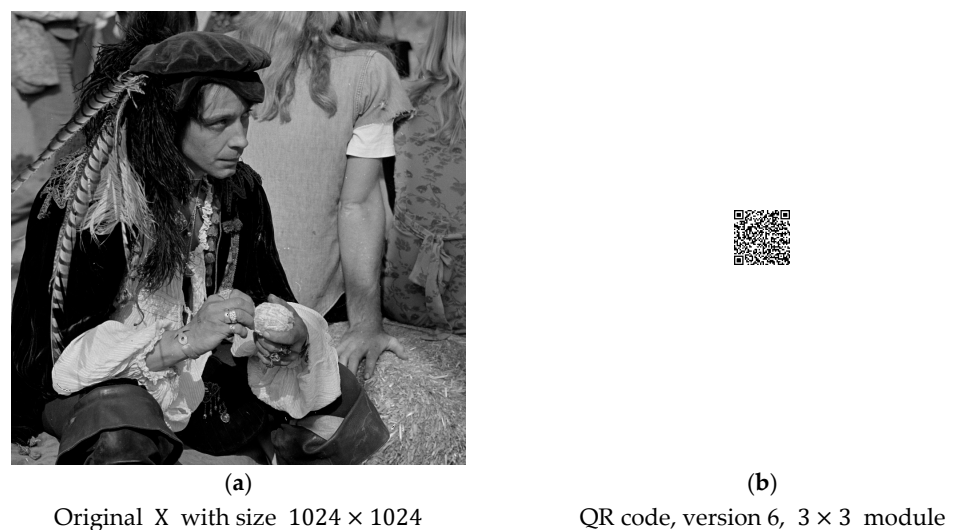


Figure 2. (a) Test image *male* with size 1024×1024 . (b) The QR code with URL for test image *male*.

For (k, n) visual secret sharing, the secret image is split into n shares, with n being a positive integer. For any combinations of k or more shares, with the constraint that $k \leq n$, they can be stacked or calculated to reconstruct the secret image. With this configuration, it is denoted by $VSS(k, n)$. For encoding with VSS, we apply the method in [14] for the generation of shares with the polynomial by using modulo operations

$$p(x) = (a_0 + a_1x + \dots + a_{k-1}x^{k-1}) \bmod p. \quad (9)$$

In Equation (9), a_0 denotes the luminance value, which lies in a range between 0 and 255, and k indicates the number of shares needed for reconstruction. The value p is a prime number larger than the maximal value of a_0 . Because the maximal value of the

luminance of the 8-bit pixel is 255, we choose the smallest possible value of $p = 257$. For $a_1, \dots, a_{k-1}$ in Equation (9), they are randomly selected numbers between 0 and $(p - 1)$. Note that the luminance value of every pixel in the share should lie between 0 and 255, so one pixel in the share is composed of 8 bits, which is identical to the grayscale image representation. There is the possibility that because $p = 257$, after calculations with Equation (9), the value may become 256, which exceeds the allowable range of the luminance values between 0 and 255. When this happens, this position should be recorded as the location map [7,14], and the calculation at such a position should be skipped. The luminance value at such a location should be kept intact.

Here is an example to demonstrate VSS for one pixel. Suppose that we are going to generate four shares with $n = 4$, and two shares with $k = 2$ are sufficient for recovery with calculations in Equation (9). If $a_0 = 160$ and $k - 1 = 1$, we may choose $a_1 = 135$ and $p = 257$. With these parameter settings, for the generation of four shares at some pixel positions, pixel values of four shares can be obtained:

$$p(1) = (160 + 135 \cdot 1) \bmod 257 = 38;$$

$$p(2) = (160 + 135 \cdot 2) \bmod 257 = 173;$$

$$p(3) = (160 + 135 \cdot 3) \bmod 257 = 51;$$

$$p(4) = (160 + 135 \cdot 4) \bmod 257 = 186.$$

After calculation, there is one value that needs to be avoided. Because $p = 257$, there is the possibility that after calculations with Equation (9), $p(x)$ may become 256, which exceeds the allowable range of the luminance values between 0 and 255. For an instance to show this kind of exception, when $a_0 = 160$, $a_1 = 96$, and $p = 257$, the luminance value at the specific position of the first share becomes 256. The luminance value at such a location should be kept intact. A location map helps to recover the secret image or the QR code. Therefore, $p = 257$ would be a suitable candidate for our setting, because when choosing $p = 263, 269, 271$, or other prime numbers with larger values, the gap between 255 and selected p values would become larger, implying an increase in the size of the location map since a larger number of locations could need to be recorded.

With the settings above, when receiving at least two shares, the pixel in the secret image should be recovered. And we can perform the verifications with Lagrange interpolation [14] based on the parameter settings of $p(1)$, $p(2)$, $p(3)$, and $p(4)$, as depicted below.

1. When we receive all four shares, or $k = 4$, we can calculate the Lagrange polynomial

$$q(x) = 38 \frac{(x-2)(x-3)(x-4)}{(1-2)(1-3)(1-4)} + 173 \frac{(x-1)(x-3)(x-4)}{(2-1)(2-3)(2-4)} + 51 \frac{(x-1)(x-2)(x-4)}{(3-1)(3-2)(3-4)} + 186 \frac{(x-1)(x-2)(x-3)}{(4-1)(4-2)(4-3)}.$$

Then, we calculate the value, $(q(0) \bmod 257)$, and obtain 160. This is identical to the pre-set value of $a_0 = 160$ above.

2. When we receive three shares, or $k = 3$, without loss of generality, we choose the first three and calculate the Lagrange polynomial

$$q(x) = 38 \frac{(x-2)(x-3)}{(1-2)(1-3)} + 173 \frac{(x-1)(x-3)}{(2-1)(2-3)} + 51 \frac{(x-1)(x-2)}{(3-1)(3-2)}.$$

Then, we calculate the value, $(q(0) \bmod 257)$, and obtain 160. This is identical to the pre-set value of $a_0 = 160$ above.

3. When we receive two shares, or $k = 2$, without loss of generality, we choose the last two and calculate the Lagrange polynomial

$$q(x) = 51 \frac{(x-4)}{(3-4)} + 186 \frac{(x-3)}{(4-3)}.$$

Then, we calculate the value, $(q(0) \bmod 257)$, and obtain 160. This is identical to the pre-set value of $a_0 = 160$ above.

4. When we receive one share, or $k = 1$, without loss of generality, we choose the second one and calculate the Lagrange polynomial to obtain $q(x) = 173$. So, $q(0) = 173$. Then, we calculate the value, $(q(0) \bmod 257)$, and obtain 173. Apparently, it is not identical to the preset value of $a_0 = 160$ above.

Based on the verification procedures above, we can apply the Lagrange polynomial to calculate the reconstructed value of $a_0 = 160$ when $k = 2, 3$, or 4 , and it can meet the requirement for VSS. The procedures above apply to one pixel position, and we can extend this concept for the whole image.

3. Motivations and Proposed Schemes

3.1. Motivations

Nowadays, people tend to use smartphone cameras to capture images. At the time of submission of this study in June 2026, the camera resolutions for conventional smartphones are around 48 to 50 MP. With this observation, when we compare with test images in the USC SIPI Image Database [12], where resolutions of test images range from 0.25 MP to 1 MP, we find the gap in resolutions between test images and the pictures captured today [13]. We suggest examining performances with our algorithm by using conventional images first and then extending this concept to check the effectiveness by using high-resolution pictures captured by ourselves. Regarding embedded watermarks, instead of using random bitstreams, we choose a QR code that conveys meaningful information. We can apply visual secret sharing and integrate VSS into our implementation with the requirement that the maximum allowable capacity can house all the VSS shares. We expect to devise a watermarking scheme that has the capability of withstanding JPEG recompression attacks. All shares are embedded into images captured by us. Extracted shares are decoded with VSS, and reconstructed QR codes from decoded VSS shares should be scannable. These are the motivations of JPEG-resistant robust watermarking and secret sharing for practical applications.

3.2. Relationships and Balances Among Performance Parameters

As mentioned in Section 2.1, performances of watermarking algorithms might be affected by the parameters selected. We can discuss their relationships in this section. Without loss of generality, to meet with 8×8 DCT, we take one 8×8 block to serve as the image block X . The concept can be extended to the whole image.

For the relationships between imperceptibility and capacity, we first observe the differences between X and X_w . When we embed 2 bits into the block, it apparently leads to larger errors, compared to embedding 1 bit into the block. This implies that larger capacities cause more levels of degradation, which has a negative effect on imperceptibility.

For the relationships between imperceptibility and robustness, it refers to which of the DCT coefficients is selected for watermark embedding and the selection of quantization stepsize Δ in QIM. For fair comparisons, we embed 1 bit into the block. From the perspective of the choice of frequency coefficient, with the embedding into the lower-frequency coefficient, it tends to withstand attacks because, when we choose JPEG attacks, it tends to discard higher-frequency coefficients, and the hidden information would also be removed. However, by altering lower-frequency coefficients due to embedding, hidden information can be kept, with the sacrifice of larger errors induced, and, hence, the imperceptibility would be adversely affected. From the perspective of quantization stepsize, when choosing a small value of Δ , small errors could be introduced due to watermarking to reach good imperceptibility. This would hardly withstand JPEG attacks for low QF values. In contrast, when selecting a large value of Δ , we may look for good robustness, with the sacrifice of significantly degraded quality in the watermarked image.

For the relationships between capacity and robustness, on the one hand, it leads to the observation that larger capacity brings more flexibility for the presentation of a watermark. For instance, when we repeatedly embed the same watermark a few times, preferably an odd number of times, we obtain the watermarked image and then the attacked image after experiencing attacks. After extraction, we may use majority vote to recover the watermark with enhanced robustness. In addition, instead of choosing random bit-streams to fill in the capacity, we may choose meaningful information, such as a QR code or VSS shares, to play the role of the watermark. On the other hand, when embedding the same amount of capacity, with an increase in quantization stepsize Δ in QIM, robustness may also be enhanced.

Summing up, to reach better imperceptibility, a smaller amount of watermark bits should be embedded into higher-frequency coefficients in the transform domain. For better robustness, a larger amount of watermark bits or a large value of quantization stepsize Δ in QIM should be embedded into lower-frequency coefficients in the transform domain. Larger capacity may lead to more flexibility for the presentation of watermark bits, with the sacrifice of more degradation to imperceptibility. Larger Δ values would also cause better robustness with the sacrifice of quality degradation. Therefore, we can observe that performance metrics are interrelated. We may find the balance among these metrics. In this study, we limit the embedding capacity by choosing to embed, at most, 1 bit per 8×8 block. Low- to middle-frequency coefficients are adjusted with the properly selected quantization stepsize to look for imperceptibility and robustness. Both the QR codes and the VSS shares are examined for the feasibility of our scheme.

3.3. Integration of Robust Watermarking with Visual Secret Sharing

Before the implementation of integration, we need to carefully choose parameters to meet the maximum allowable capacity. For the QR code, its resolution directly relates to its version number. For VSS, the numbers of (k, n) should be selected. Every pixel in each share is represented by 8 bits, implying the value can range between 0 and 255. Every share has the same resolution as the QR code, and it directly follows the version of the QR code and the module size selected.

For a conventional 1024×1024 test image, with our setting, the maximally allowable capacity is 16384 bits. Considering presentations with VSS(k, n) shares, resolutions of the total n shares should be less than or equal to $\frac{16,384}{8} = 2048$ pixels. Suppose that we use the $t \times t$ module in a version v QR code to meet the resolutions of $((17 + 4v) \cdot t) \cdot ((17 + 4v) \cdot t) \cdot n \leq 2048$. For VSS, n should be at least 2. With these settings, it becomes $((17 + 4v) \cdot t) \cdot ((17 + 4v) \cdot t) \leq 1024$, or $(17 + 4v) \cdot t \leq 32$. Therefore, only $t = 1$, or 1×1 module, and $v = 1, 2, 3$ are satisfied. With these backgrounds, VSS would be hardly applicable for conventional test images. With the solutions for the parameters above, we have three conditions: (a) use of QR codes with versions 1, 2, or 3; (b) generation of shares for VSS(2, 2) or VSS(1, 2); and (c) selection of 1×1 module in the QR codes. When the three conditions are all satisfied, they become the possible settings under extreme cases. When we choose the 2×2 module for the QR code, VSS becomes impractical for conventional test images.

In contrast, for 8192×6144 images, with our setting, the maximally allowable capacity is 786,432 bits. Resolutions of the total n shares should be less than or equal to $\frac{786,432}{8} = 98,304$ pixels. To meet with the requirement in VSS, we turn the binary QR code into 8-bit presentation. The black and white pixels in the binary QR code can be assigned to luminance values of 0 and 255, respectively. Subsequently, VSS shares can be calculated with Equation (9).

As we pointed out in Section 3.3, when we choose the three conditions for evaluating embedding capacity, namely, (a) use of QR codes with version v , (b) generation of n shares, and (c) selection of $t \times t$ module, the embedding capacity of all shares becomes $(17 + 4v)^2 \cdot n \cdot t^2$ bytes of resolution or a total of $(17 + 4v)^2 \cdot n \cdot t^2 \cdot 8$ bits. In this manner, $(17 + 4v)^2 \cdot n \cdot t^2 \cdot 8 \leq 786,432$, or $((17 + 4v) \cdot t)^2 \cdot n \leq 98,304$. The number of shares, n , should be at least 2 to meet the requirements in VSS. Moreover, with the consideration of enhanced robustness, we perform majority vote of pixel values inside every module, and this leads to $t \geq 2$.

Regarding majority vote, we would prefer to see the module with odd-numbered resolutions. For instance, for the 3×3 module, after the extraction of bits from the attacked image and decoding with VSS, if we have five or more luminance values that are larger than or equal to 128, all nine pixels in such a module are represented with 255. Otherwise, they are represented with 0. However, even-numbered resolutions also exist, and there is the possibility that half of the pixels in the module are larger than or equal to 128, and the other half are smaller than 128. For instance, for the 2×2 module, for the convenience of applying the concept of majority vote for our implementation, if we have two or more luminance values that are larger than or equal to 128, all four pixels in such a module are represented with 255. The only problem is that when two luminance values are smaller than 128 and two are not, where $C_2^4 = 6$ out of $2^4 = 16$ combinations, or 37.50%, would meet this dilemma, we directly assign all four pixels to 255 for convenience. When the module size becomes larger, for instance, for 4×4 module, $C_8^{16} = 12870$ out of $2^{16} = 65,536$ combinations, or 19.64%, would meet this dilemma. Larger module size would decrease the ratio for even-numbered modules. Considering the robustness of JPEG attacks, simulations present the effectiveness with our implementation.

The inequality mentioned above offers the rule of thumb for the selection of version number and module size. Table 1 depicts the suitable selections of parameters between two shares and eight shares in VSS.

Table 1. Rule of thumb selection for share number, module size, and QR code version for the generation of VSS shares that fit the maximal capacity in robust watermarking.

Shares, n	Module Size, $t \times t$	QR Code Version, v
$n = 2$	2×2	$v = 1, \dots, 23$
	3×3	$v = 1, \dots, 14$
	4×4	$v = 1, \dots, 9$
	5×5	$v = 1, \dots, 6$
	6×6	$v = 1, \dots, 5$
	7×7	$v = 1, 2, 3$
	8×8	$v = 1, 2$
	9×9	$v = 1$
$n = 3$	2×2	$v = 1, \dots, 18$
	3×3	$v = 1, \dots, 10$
	4×4	$v = 1, \dots, 7$
	5×5	$v = 1, \dots, 4$
	6×6	$v = 1, 2, 3$
	7×7	$v = 1, 2$
	8×8	$v = 1$
$n = 4$	2×2	$v = 1, \dots, 15$
	3×3	$v = 1, \dots, 8$
	4×4	$v = 1, \dots, 5$
	5×5	$v = 1, 2, 3$
	6×6	$v = 1, 2$
	7×7	$v = 1$
$n = 5$	2×2	$v = 1, \dots, 13$

	3×3	$v = 1, \dots, 7$
	4×4	$v = 1, \dots, 4$
	5×5	$v = 1, 2$
	6×6	$v = 1$
$n = 6$	2×2	$v = 1, \dots, 11$
	3×3	$v = 1, \dots, 6$
	4×4	$v = 1, 2, 3$
	5×5	$v = 1, 2$
	6×6	$v = 1$
$n = 7$	2×2	$v = 1, \dots, 10$
	3×3	$v = 1, \dots, 5$
	4×4	$v = 1, 2, 3$
	5×5	$v = 1$
$n = 8$	2×2	$v = 1, \dots, 9$
	3×3	$v = 1, \dots, 5$
	4×4	$v = 1, 2$
	5×5	$v = 1$

With the massive amount of capacity, we have flexible selections for QR codes and the number of shares in VSS. According to Table 1, in our simulations, we choose two types of parameters:

- Seven shares in VSS(5, 7), 2×2 module, for version 10 QR code. Number of embedding pixels equals $((17 + 40) \cdot 2) \cdot ((17 + 40) \cdot 2) \cdot 7 = 90,972$. And it lies in the range of 98,304 pixels.
- Three shares in VSS(2, 3), 3×3 module, for version 10 QR code. Number of embedding pixels equals $((17 + 40) \cdot 3) \cdot ((17 + 40) \cdot 3) \cdot 3 = 87,723$. And it lies in the range of 98,304 pixels.

We can see that due to the increase in resolutions, we have lots of flexibility for the selection of parameters, as depicted in Table 1, that makes hiding VSS shares with watermarking a feasible solution.

3.4. Security Profile for Proposed Framework

We would like to discuss a more explicit, formalized security profile for the proposed framework. To clarify the security posture of the proposed framework, we explicitly define our security model below.

1. Threat model. We assume a standard Dolev–Yao-style channel adversary capable of passive interception and active channel manipulation [29]. The passive adversary aims to extract unauthorized QR data. The active adversary aims to disrupt authentication via lossy JPEG reprocessing.
2. Cryptographic key management. Security relies on a two-tier key framework. The first is the symmetric PRNG seed acting as the secret coordinate mapping key for coefficient selection between AC3 and AC14 in Figure 1. And the second is ephemeral random coefficients of a_1 to a_{k-1} in Equation (9) acting as localized one-time-pad equivalents in the polynomial space $\mathbb{Z}_{257}$.
3. Proof against share leakage. The underlying polynomial secret sharing provides information-theoretic security. As mathematically validated in Section 2.3, any coalition of attackers capturing fewer than k shares receive exactly 0 bits of mutual information regarding the underlying QR code, rendering collusion or partial leakage mathematically harmless.
4. Attack complexity and disruption boundary. For an attacker to blindly strip the watermark without knowledge of the secret PRNG key, they must apply extreme JPEG quantization for low QF values. Countering this requires scaling QIM stepsize Δ in

our framework, forcing the attacker into a dilemma: they can only disrupt our watermark by executing an attack that renders the host image visually worthless due to severe blocking artifacts.

In addition to the motivation, balance of performance parameters, and integration of watermarking and QIM, we also provide the security profile for better comprehension of the proposed framework.

4. Simulation Results

Here, we choose 15 images in four categories in our simulations. They are briefly explained as follows.

- The first category contains eight test images with sizes of 512×512 . These are airplane, baboon, Barbara, boat, coronet, goldhill, peppers, and truck [12].
- The second category contains one image, male [12], which has a size of 1024×1024 .
- The third category contains one image, lake, captured by the authors, which has a size of 8192×6144 .
- The fourth category contains five high-resolution images, bedroom, castle, dog, ducks, and moon, obtained from Unsplash website with no copyright issues [13].

The images in the first and second categories are conventional test images to verify the effectiveness and feasibility of our scheme and to compare results in the literature. The images in the third and fourth categories are high-resolution images used to test the practicality of our scheme. Our goal is to keep the maximally allowable capacity for watermark embedding, to observe marked image quality, and to examine the robustness of the extracted watermark, including both the extracted QR code and reconstructed QR code decoded from the composition of VSS shares.

The simulations and discussions are organized as follows. In the first category simulations, for fair comparisons with results in [19–22], we keep the same conditions for performing experiments. We pick each of the eight test images, set the same amount of embedding capacity, and choose the random bitstreams of 1024 bits to serve as the watermark. After completing the experiments, we discuss the watermarked image quality and robustness with several quality factors in JPEG attacks.

In the second category simulations, on the male image, we have $\frac{1024}{8} \times \frac{1024}{8} = 16,384$ non-overlapping blocks. We choose to embed 1 bit per block to check the feasibility of our method. Hence, we may embed, at most, 16,384 bits. Considering the requirements with QR codes, we can generate information corresponding to male image and choose version 5, which has 37×37 modules, with a module size of 3×3 . In this way, the QR code becomes a bi-level image with size 111×111 , or 12,321 bits.

In the third category simulations, on the lake image, we have $\frac{8192}{8} \times \frac{6144}{8} = 786,432$ non-overlapping blocks. We choose to embed 1 bit per block of VSS shares to check the flexibility of our method. Hence, we may embed, at most, 786,432 bits, which leads to 48-times the embedding capacity compared to its counterpart with the male image. Due to the abundant amount of capacity, we employ visual secret sharing for implementation and hide secret shares in the image. With the descriptions in Section 3.3, we choose two types of VSS for embedding. The generation of a QR code is based on the geographical coordinates where the image was captured. We intentionally choose a lengthy URL that corresponds to the QR code with a larger version number in our simulations. The URL is as follows:

<https://maps.google.com/maps?ll=35.517778,138.756111&q=35.517778,138.756111&hl=ja&t=m&z=12>.

In the fourth category simulations, our method is applied to the five public test images. Statistics about the performances will be discussed to support the practicality of our scheme.

The first category simulations are designed to verify the feasibility of our scheme. We compare the performance metrics of watermarked image quality in the peak signal-to-noise ratio (PSNR) and structural similarity (SSIM), capacity in bits, and robustness in bit correct rate (BCR), with peppers image shown in Figure 3, which has a size of 512×512 , along with seven other test images, which also have sizes of 512×512 , from the USC-SIPI Database [12]. The test images in Figures 2a and 3 are scaled proportionally for better comprehension. Note that in [19–22], instead of choosing male image in Figure 2a, the authors selected peppers, airplane, baboon, Barbara, boat, coronet, goldhill, and truck, which are a quarter as big as the former one. For fair comparisons, we also choose all eight images to present the performance metrics. In addition, in [19–22], they only presented results with the eight images in the first category, and there is not enough information to make a valid comparison with the male image. In [19–22], the authors employed correlations in 2×2 neighboring blocks, and only 1 bit can be embedded into four blocks. In this way, a total of 1024 bits can be embedded. Our scheme can meet this condition, from embedding 4096 bits to 1024 bits. We can also embed the reduced capacity of 1 bit in four blocks. In Tables 2 and 3, we present the imperceptibility in PSNR and SSIM, respectively. Our scheme performs best in most cases. Under the condition of the same embedding capacity, our results outperform those in [19,20,22] in imperceptibility represented by PSNR and SSIM. Our results perform better than those in [21], except for nearly the same result in PSNR and SSIM.

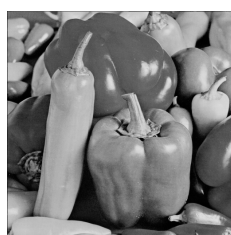


Figure 3. Test image peppers with size 512×512 . Figures 2a and 3 are scaled proportionally according to their sizes.

Table 2. Comparisons of imperceptibility in PSNR with eight test images with size 512×512 .

Quality	Proposed	In [19]	In [20]	In [21]	In [22]
airplane	44.27	42.93	44.07	42.05	41.47
baboon	45.64	40.87	44.04	40.14	38.92
Barbara	44.68	42.26	41.55	42.26	41.91
boat	44.38	44.52	44.87	45.24	43.36
coronet	44.34	44.07	46.74	43.07	44.45
goldhill	44.10	42.67	43.39	43.04	43.39
peppers	44.32	43.36	44.88	42.21	42.54
truck	45.15	42.67	44.26	42.25	43.71

Table 3. Comparisons of imperceptibility in SSIM with eight test images with size 512×512 .

Quality	Proposed	In [19]	In [20]	In [21]	In [22]
airplane	0.9818	0.9867	0.9880	0.9722	0.9805
baboon	0.9984	0.9836	0.9962	0.9805	0.9918
Barbara	0.9908	0.9798	0.9892	0.9798	0.9872
boat	0.9850	0.9876	0.9876	0.9821	0.9801
coronet	0.9839	0.9828	0.9877	0.9792	0.9979
goldhill	0.9825	0.9763	0.9874	0.9786	0.9866

peppers	0.9869	0.9823	0.9886	0.9753	0.9828
truck	0.9953	0.9845	0.9888	0.9827	0.9876

In Tables 4 and 5, we present the robustness under JPEG attacks in BCR and NC, respectively. The averages of BCR and NC with the eight test images in Tables 2 and 3 are displayed. Under some test conditions for robustness, corresponding results in the references [19–22] are not provided, and, hence, we use the abbreviation NA, implying ‘not provided’. Under the condition of the same embedding capacity, our results outperform those in [19–22] in robustness to JPEG attacks represented by BCR and NC. In brief, our scheme should be feasible for extension to high-resolution images due to the better performance on imperceptibility and robustness.

Table 4. Comparisons of robustness in BCR with the average of eight test images with size 512×512 .

Robustness	Proposed	In [19]	In [20]	In [21]	In [22]
QF = 10	41.75%	NA	NA	NA	NA
QF = 20	99.85%	NA	NA	NA	NA
QF = 60	100.00%	99.86%	99.06%	98.21%	93.20%
QF = 70	100.00%	99.94%	99.90%	99.67%	92.48%
QF = 80	100.00%	100.00%	99.97%	100.00%	92.64%

Table 5. Comparisons of robustness in NC with the average of eight test images with size 512×512 .

Robustness	Proposed	In [19]	In [20]	In [21]	In [22]
QF = 10	0.5830	NA	NA	NA	NA
QF = 20	0.9989	NA	NA	NA	NA
QF = 60	1.0000	0.9973	0.9903	0.9805	0.9320
QF = 70	1.0000	0.9991	0.9994	0.9952	0.9252
QF = 80	1.0000	1.0000	0.9996	1.0000	0.9268

In JPEG compression, the quality factor (QF) ranges from 1 to 100. It determines the scaling of quantization tables, directly dictating the balance between file size and visual fidelity. Setting the QF between 1 and 20 represents extreme compression. This range yields very small file sizes but causes massive and often unacceptable degradation in image quality. With JPEG attacks using a QF less than 20 on a watermarked image, the attacked image would lead to severe blocking effects, loss of sharp edges or textures, and ringing artifacts [18]. It serves as a rigorous stress-test mechanism to evaluate the ultimate survivability and boundaries of robust watermarking algorithms. Testing at QF less than 20 helps map out the exact failure threshold of a watermarking scheme, proving whether it can withstand extreme, adversarial conditions. For comprehensive tests of JPEG attacks to high-resolution images with low QF values, we also evaluate these extreme cases in category 3 and 4 simulations.

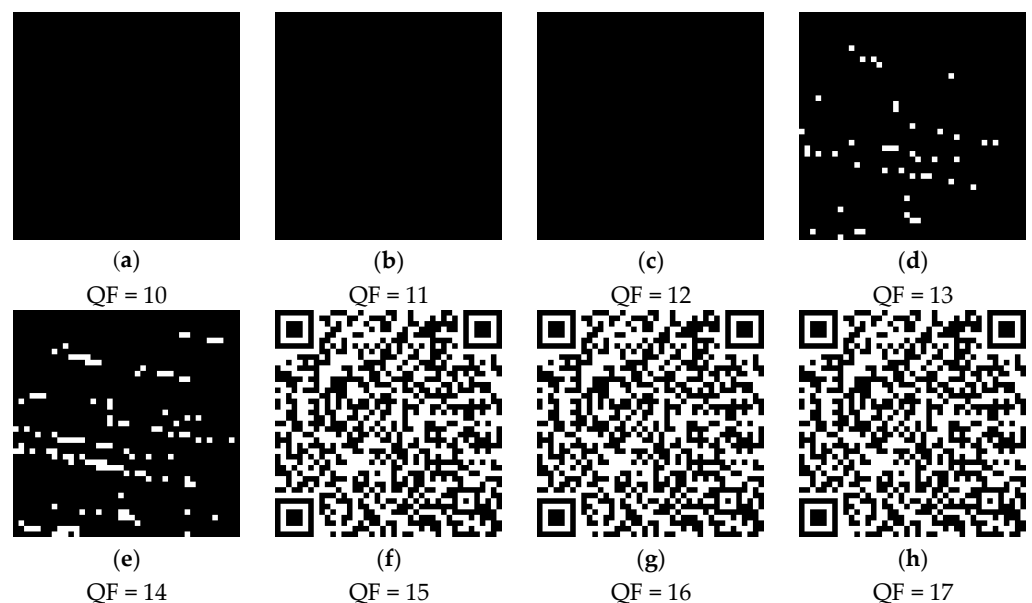
In the second category, Figure 4 displays the watermarked image after embedding the QR code in Figure 2b into the original image in Figure 2a. The QR code in Figure 2b has 15,129 bits, which is under the maximally allowable capacity of 16,384 bits in the male image. We calculate the DCT coefficients, select the embedding coefficient in the range between AC3 and AC14 by using PRNG, and perform embedding with QIM. After embedding, we calculate inverse DCT and obtain the watermarked image in Figure 4. Regarding imperceptibility, Figure 4 has a PSNR of 39.37 dB and SSIM of 0.9584. Due to the limited capacity, we only embed the binary QR code into the conventional test image.



Figure 4. Watermarked image male after embedding Figure 2b into Figure 2a. PSNR= 39.37 dB. SSIM= 0.9584.

Figure 5 presents the watermarks extracted from the attacked male image after experiencing JPEG attacks. Detailed values for BCR are depicted in Table 6. In Figure 5a–e, which correspond to results after JPEG attacks with QF from QF = 10 to QF = 14, they cannot be scannable, and the BCR values are far from satisfactory. In Figure 5f–h, even though BCR values are not 100.00% extracted watermarks can be scannable. In Figure 5i–l, the BCR values reach 100.00%. For QF values larger than 30, they act the same as Figure 5l till QF = 100.

With the results in Figure 5 and Table 6, our scheme can survive JPEG attacks from QF = 15 to QF = 100. It fails to survive for QF = 14 or smaller. As mentioned above, the watermarked image attacked by using QF less than 20 would cause blocking effects or loss of sharp edges. It would be rarely seen except for providing stress tests across all the possible QF values. We also observed that in [19–22], the authors only provided results with QF = 60 or larger, which may serve as a reference for a reasonable QF value for JPEG attacks. Therefore, we can claim that the proposed scheme is JPEG-resistant.



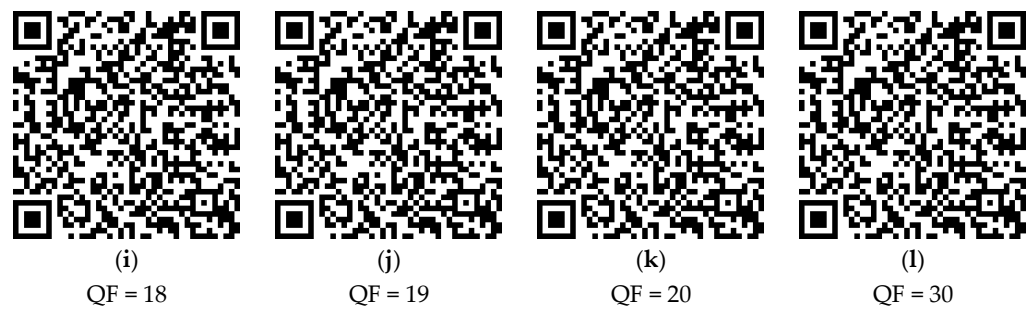


Figure 5. Extracted watermarks from attacked male image after experiencing JPEG attacks with different quality factors.

Table 6. Robustness in bit correct rate (BCR) with male image by using JPEG attack with different quality factors (QF).

JPEG Attack	BCR	Scannable?
QF = 10	48.72%	No
QF = 11	48.72%	No
QF = 12	48.72%	No
QF = 13	51.28%	No
QF = 14	55.44%	No
QF = 15	99.94%	Yes
QF = 16	99.88%	Yes
QF = 17	99.94%	Yes
QF = 18	100.00%	Yes
QF = 19	100.00%	Yes
QF = 20	100.00%	Yes
QF = 30	100.00%	Yes
QF = 40	100.00%	Yes
QF = 100	100.00%	Yes

In the third category, Figure 6 displays the watermarked image after embedding all seven VSS shares with the setting of VSS(5, 7) in Figure 7. Each pixel in the seven VSS shares is represented by 8-bit due to the requirement in secret sharing. These shares are calculated from the QR code with a module size of 2×2 , leading to a total of 727,776 bits, and this amount is smaller than the maximally allowable capacity of 786,432 bits in the lake image. We calculate the DCT coefficients, select the embedding band in the range between AC3 and AC14 by using PRNG, and perform embedding with QIM. After embedding, we calculate inverse DCT and obtain the watermarked image in Figure 6. Regarding the imperceptibility, Figure 6 has a PSNR of 38.95 dB and SSIM of 0.9147, after embedding 727,776 bits. Due to the increased capacity, we can embed all seven shares in Figure 7 into the image captured by ourselves.



Figure 6. Watermarked image lake with resolutions of 8192×6144 . PSNR = 38.95 dB. SSIM = 0.9147. Note that the sizes between Figures 2a, 3, and 6 are not presented at the same scale due to the large resolutions of Figure 6.

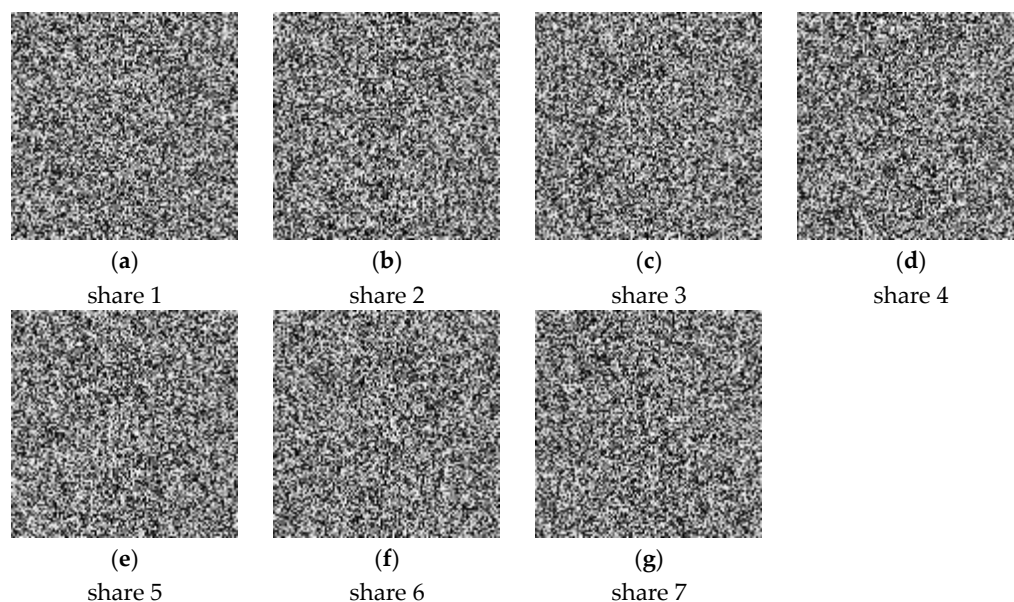


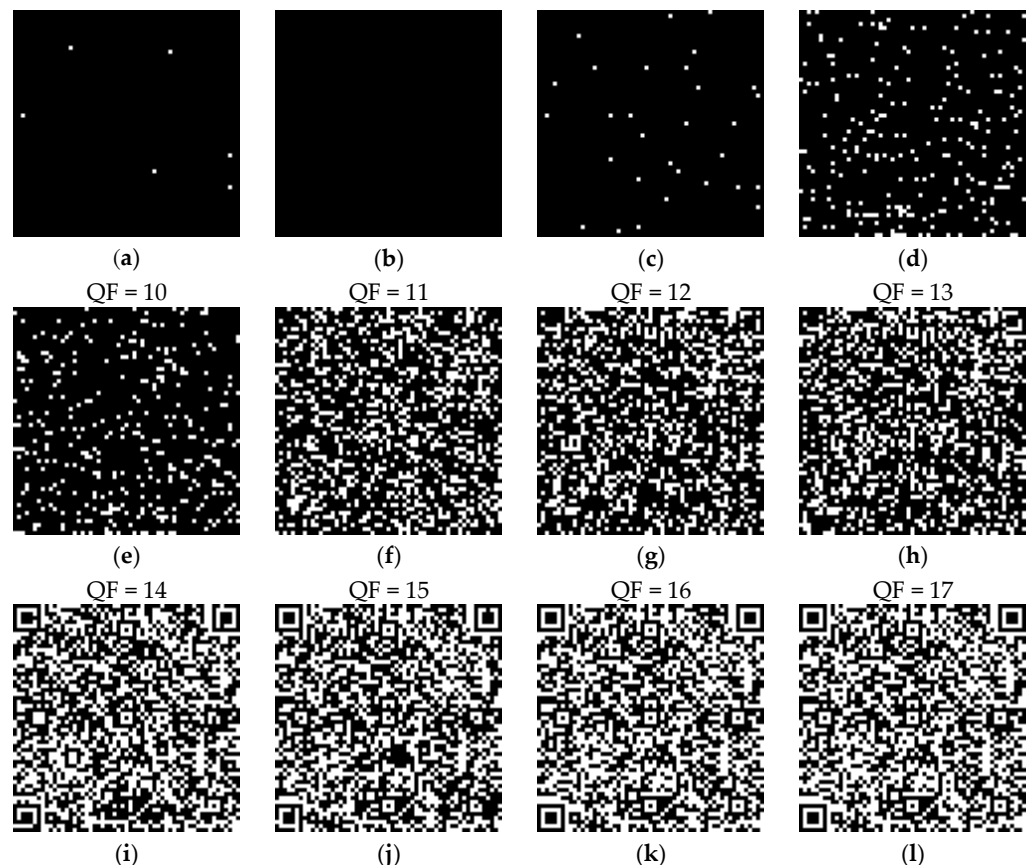
Figure 7. Seven shares in VSS(5, 7). Any five of them can compose the reconstructed QR code.

Figure 8 presents the reconstructed watermark after obtaining any five of the seven shares in Figure 7 when no attack is applied. We first perform watermark extraction with QIM to acquire the shares and apply VSS decoding with the procedures in Section 2.3. Subsequently, in one module with a size of 2×2 , the four luminance values are calculated. With the majority vote described in Section 3.3, the corresponding luminance value of 0 or 255 is placed into the 2×2 module. It is scannable with a smartphone camera or other apps to capture and recognize the QR code. In Figure 9, we extract watermark bits from the attacked lake image, compute the reconstructed QR code with VSS decoding,

and perform majority vote. Corresponding to Figure 9, BCR values, and whether the reconstructed QR code is scannable, are also depicted in Table 7. We observe low BCR values when $QF = 14$ or smaller, implying the extremes that our scheme is incapable of withstanding JPEG attacks with small QF values. When $QF = 15$ to 18 , we can see higher BCR values; however, the corresponding QR codes are not scannable. This might occur for several reasons. First, because the finder patterns can hardly be recoverable due to attacks, it makes the smartphone camera or apps unable to recognize the information therein. Second, we use the smartphone to scan the QR code on the screen even when the repeated trials have been executed. Glossy laminated glass reflects light directly into the camera lens. And, finally, the URL in Figure 8 is too long. We would like to verify the feasibility of our scheme by using such a long URL. A long character sequence directly results in a larger version of the QR code. For practical implementation, a URL shortener may be applied. When $QF = 19$ or larger, reconstructed QR codes are scannable, even though some BCR values do not reach 100%, as depicted in Table 7.



Figure 8. Reconstructed watermark after obtaining any five shares in Figure 7 and calculating with VSS decoding in Section 2.3.



QF = 18

QF = 19

QF = 20

QF = 30

Figure 9. Extracted watermarks or reconstructed QR codes, calculated by extracted shares, from attacked lake image after experiencing JPEG attacks with different quality factors. It needs at least five shares to reconstruct the QR code.

Table 7. Robustness in bit correct rate (BCR) with lake image for VSS(5, 7) by using JPEG attack with different quality factors (QF).

JPEG Attack	BCR	Scannable?
QF = 10	50.12%	No
QF = 11	50.20%	No
QF = 12	50.40%	No
QF = 13	50.92%	No
QF = 14	51.93%	No
QF = 15	96.67%	No
QF = 16	97.78%	No
QF = 17	97.93%	No
QF = 18	99.15%	No
QF = 19	99.54%	Yes
QF = 20	99.85%	Yes
QF = 30	99.92%	Yes
QF = 40	100.00%	Yes
QF = 100	100.00%	Yes

Compared to the results in Figure 5 and Table 6, which relate to conventional test image male, the BCR values perform well when QF = 15 or larger in both cases. When we embed the QR code in male, the extracted QR code is scannable when QF = 15 or larger, which follows the trends in BCR values. When we apply VSS to add another level of security, the reconstructed QR code can be scannable for QF = 19 or larger. Even if the BCR values between QF = 15 and 18 perform well, the reconstructed QR code is unable to be scanned due to the characteristics of the QR code.

Figure 10 presents all three VSS shares with the setting of VSS(2, 3). Each pixel in the three VSS shares is represented by 8-bit due to the requirement in secret sharing. These shares are calculated from the QR code with a module size of 3×3 , leading to a total of 701,784 bits, and this amount is smaller than the maximally allowable capacity of 786,432 bits in lake image. We calculate the DCT coefficients, select the embedding band in the range between AC3 and AC14 by using PRNG, and perform embedding with QIM.

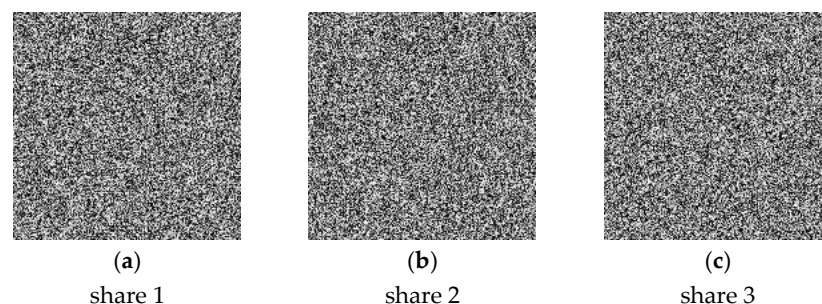


Figure 10. Three shares in VSS(2, 3). Any two of them can compose the reconstructed QR code.

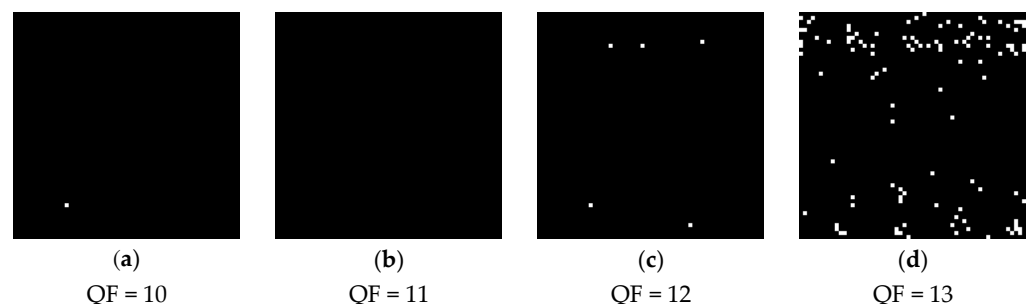
After embedding, we calculate inverse DCT and obtain a watermarked image. Due to the massive file size in the marked image, we skipped demonstrating the watermarked image with this setting. Regarding the imperceptibility, the watermarked image with this setting has a PSNR of 39.10 dB and SSIM of 0.9162, after embedding 701,784 bits. Due to the increased capacity, we can embed all three shares in Figure 10 into the image

captured by us. In Figures 7 and 10, other than the settings of VSS(5, 7) and VSS(2, 3), respectively, the major difference is the selection of the module size in the QR codes. With the prerequisites in Table 1 to meet the maximally allowable capacity for embedding, in Figure 7, we choose 2×2 modules, which lead to the embedding of seven shares at most. The number of received shares, five shares in this case, can be chosen by algorithm designers. Correspondingly, in Figure 10, we choose 3×3 modules, which leads to the embedding of three shares at most. Again, the number of received shares, two shares in this case, can be chosen by algorithm designers.

Figure 11 presents the reconstructed watermark after obtaining any two of all three shares. We first perform watermark extraction with QIM to acquire the shares and apply VSS decoding with the procedures in Section 2.3. Subsequently, in one module with size 3×3 , we acquire the nine luminance values. With majority vote, a corresponding luminance value of 0 or 255 is placed into the 3×3 module. It is scannable with a smartphone camera or other apps to capture and recognize the QR code. In Figure 12, we extract watermark bits from the attacked lake image, compute the reconstructed QR code with VSS decoding, and perform majority vote within pixels in one module. Corresponding to Figure 12, BCR values, and whether the reconstructed QR code is scannable, are also depicted in Table 8. We observe low BCR values when $QF = 14$ or smaller, implying the extremes to withstand JPEG attacks with our scheme. When $QF = 15$ to 18, we can see higher BCR values; however, corresponding QR codes are not scannable. This might be because the finder patterns can hardly be recoverable due to attacks, and, hence, the smartphone camera or apps are unable to recognize the information therein. When $QF = 19$ or larger, reconstructed QR codes are scannable, even though some BCR values do not reach 100%, as depicted in Table 8. We also notice similar phenomena in Table 7.



Figure 11. Reconstructed watermark after obtaining any two shares in Figure 10 and calculating with VSS decoding in Section 2.3.



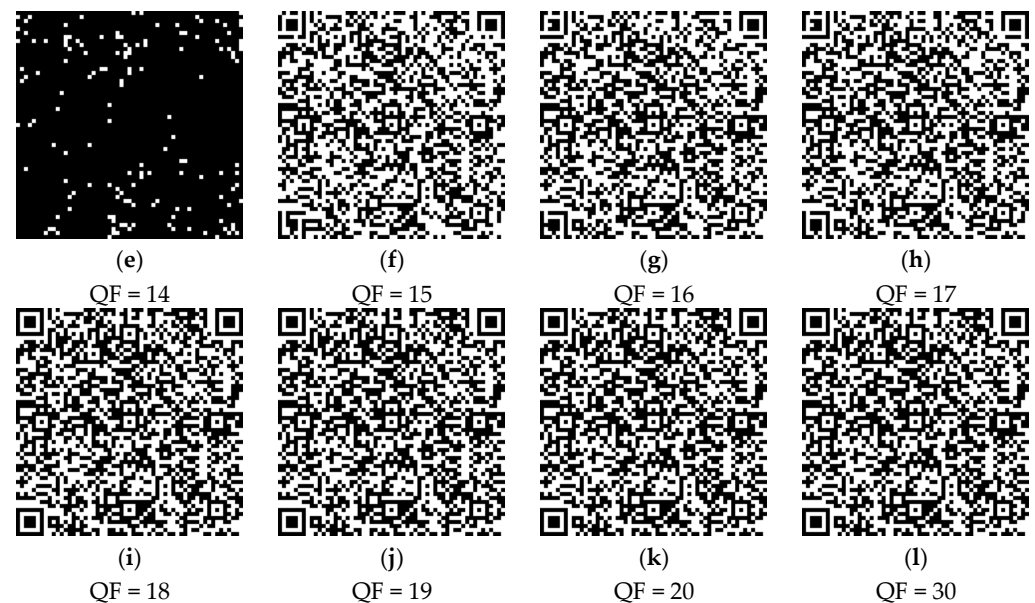


Figure 12. Extracted watermarks or reconstructed QR codes, calculated by extracted shares, from attacked lake image after experiencing JPEG attacks with different quality factors. It needs at least two shares to reconstruct the QR code.

Table 8. Robustness in bit correct rate (BCR) with lake image for VSS(2,3) by using JPEG attack with different quality factors (QFs).

JPEG Attack	BCR	Scannable?
QF = 10	50.30%	No
QF = 11	50.38%	No
QF = 12	50.56%	No
QF = 13	51.02%	No
QF = 14	52.01%	No
QF = 15	96.86%	No
QF = 16	97.98%	No
QF = 17	98.11%	No
QF = 18	99.24%	Yes
QF = 19	99.60%	Yes
QF = 20	99.85%	Yes
QF = 30	99.93%	Yes
QF = 40	100.00%	Yes
QF = 100	100.00%	Yes

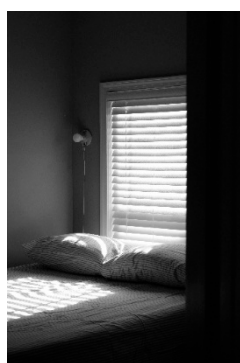
In Tables 6–8, we observe that there is a gap for BCR values between QF = 14 and QF = 15. Therefore, we would like to look for the possibility of withstanding JPEG attacks with all QF values regardless of the watermarked image quality. With Equation (6), we may increase the stepsize Δ in QIM to look for robustness for the 100 QF values under JPEG attacks. To save space for presentation, based on the statistics in Tables 6–8, we display the simulation results between QF = 1 and QF = 20 in Table 9. We can see clearly that to achieve robustness for all QF values, we need to sacrifice the watermarked image quality. Based on Figure 10, a total of 701,784 bits can be embedded to prepare for the statistics in Table 9. The top priority is to acquire the extracted watermark, which is scannable, with a high BCR value. With the increase in Δ , the proposed method is feasible to be JPEG-resistant.

Table 9. Robustness of JPEG-resistant watermarking with lake image for very low QF values.

JPEG Attack	Δ in QIM	PSNR (dB)	SSIM	BCR
QF = 1	17.5	15.99	0.5034	99.98%
QF = 2	17.0	16.30	0.5046	99.99%
QF = 3	11.5	20.56	0.5269	100.00%
QF = 4	8.5	23.51	0.5556	99.97%
QF = 5	7.0	23.53	0.5852	100.00%
QF = 6	5.8	26.96	0.6112	100.00%
QF = 7	5.0	28.27	0.6391	99.99%
QF = 8	4.8	28.63	0.6474	99.99%
QF = 9	4.4	29.40	0.6658	99.99%
QF = 10	4.1	30.03	0.6815	100.00%
QF = 11	3.4	31.69	0.7252	99.98%
QF = 12	3.1	32.51	0.7475	99.98%
QF = 13	2.9	33.10	0.7635	99.98%
QF = 14	2.7	33.74	0.7807	99.98%
QF = 15	2.5	34.43	0.7989	99.97%
QF = 16	2.3	35.17	0.8181	99.97%
QF = 17	2.2	35.53	0.8235	99.96%
QF = 18	2.1	35.99	0.8382	99.96%
QF = 19	2.0	36.42	0.8483	99.96%
QF = 20	1.9	38.95	0.9147	99.85%

From practical viewpoints, when we select appropriate values of Δ , as those in Tables 6–8 for example, watermarked image quality is good, while the method may hardly withstand JPEG attacks of QF = 15 or less. The proposed scheme demonstrates robustness over a wide range of JPEG quality factors. In contrast, it still has the capability of withstanding attacks for all QF values by intentionally increasing the values of Δ with the sacrifice of much degraded quality of marked images. The user may determine the balance between the level of robustness and the quality of the watermarked image.

Finally, in the fourth category, for the verification of the effectiveness of our scheme for images with high resolutions, in addition to the lake image captured by ourselves, we also include five diverse images, which also have large resolutions and different characteristics, from Unsplash website [13] with no copyright issues. The five selected images are displayed in Figure 13, scaled proportionally to their original dimensions. The resolutions of the images range from 8.72 MP to 94.83 MP to meet a wide range of scalability. Due to high resolutions for the representation of images, images in Figure 13 and their counterparts in Figures 4 and 6 are not at the same scaling factor.



(a)

bedroom, size 4160 × 6240



(b)

castle, size 4000 × 6000

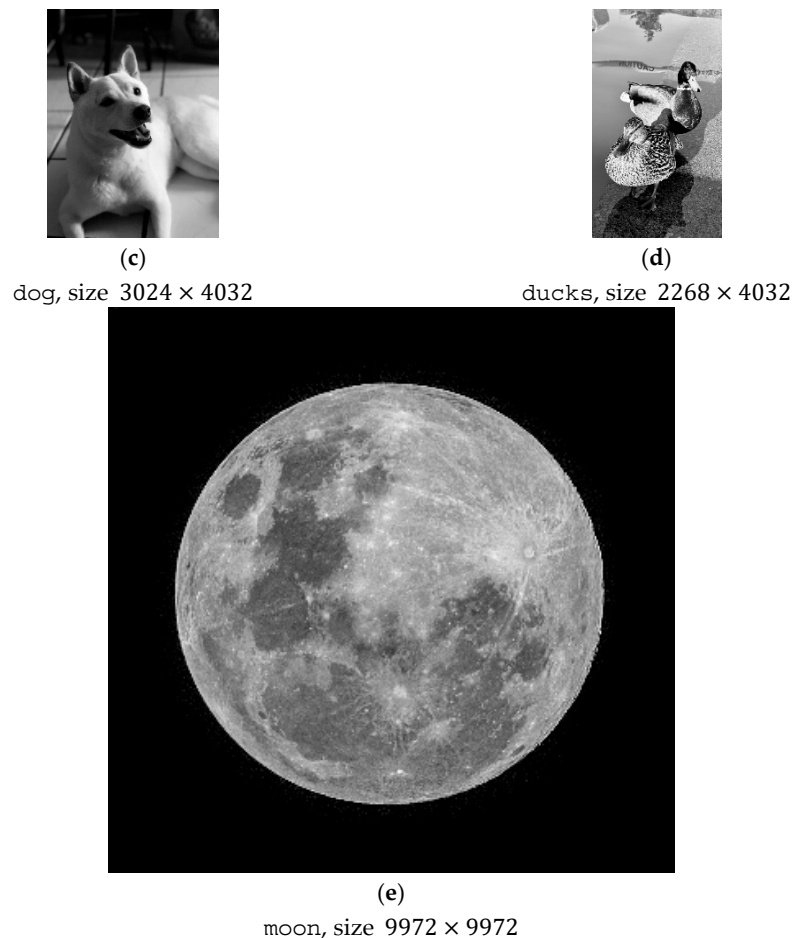


Figure 13. Five high-resolution images from Unsplash website with the resolutions shown. Note that five images are downsampled while preserving their original relative aspect ratios and sizes.

We performed experiments on the five newly added images, along with the lake image captured by ourselves, to present the feasibility with our scheme. Performance metrics are provided in Tables 10–13. In Table 10, we present the imperceptibility of these images. With the proper adjustment of Δ values between 1.9 and 2.6, we have good quality in PSNR and SSIM. The PSNR values are at least 38 dB to present good quality. The SSIM, except for moon, is at least 0.9. This might be because there are dark backgrounds in this image. Even though the PSNR is good, the structural similarity may perform a bit differently. Our scheme works well for all six high-resolution images with diversity in characteristics.

Table 10. General comparisons of imperceptibility for high-resolution images.

Image	Lake	Bedroom	Castle	Dog	Ducks	Moon
Image size	8192 × 6144	4160 × 6240	4000 × 6000	3024 × 4032	2268 × 4032	9972 × 9972
Imperceptibility						
PSNR (dB)	39.10	38.73	38.65	39.30	38.49	38.49
SSIM	0.9162	0.9085	0.9184	0.9191	0.9380	0.8373
Δ in QIM	1.9	2.3	2.6	2.6	2.6	2.3

Table 11. General comparisons of watermark composition and corresponding capacity for high-resolution images.

Image	Lake	Bedroom	Castle	Dog	Ducks	Moon
Image size	8192 × 6144	4160 × 6240	4000 × 6000	3024 × 4032	2268 × 4032	9972 × 9972

QR code version	Watermark composition					
	10	5	5	6	6	10
Module size	3 × 3	3 × 3	3 × 3	3 × 3	3 × 3	3 × 3
VSS	(2, 3)	(2, 4)	(2, 3)	(1, 1)	(1, 1)	(4, 6)
Capacity						
Capacity (bit)	701,784	394,272	295,704	121,032	121,032	1,403,568
Capacity ratio	0.01394	0.01519	0.01232	0.00993	0.01324	0.01411

Table 12. General comparisons of robustness to JPEG attacks under different QF for high-resolution images.

Image	Lake	Bedroom	Castle	Dog	Ducks	Moon
Image size	8192 × 6144	4160 × 6240	4000 × 6000	3024 × 4032	2268 × 4032	9972 × 9972
Robustness to JPEG attacks under different QF						
QF = 10	50.30%	50.11%	50.45%	49.23%	49.21%	50.13%
QF = 11	50.38%	50.13%	52.08%	50.71%	51.41%	50.18%
QF = 12	50.56%	50.71%	53.77%	53.83%	53.56%	50.66%
QF = 13	51.02%	54.25%	91.90%	92.26%	90.07%	53.98%
QF = 14	52.01%	56.72%	97.77%	99.81%	94.82%	56.93%
QF = 15	96.86%	97.21%	97.86%	99.84%	95.70%	83.26%
QF = 16	97.98%	98.04%	99.27%	99.96%	98.03%	85.27%
QF = 17	98.11%	98.55%	99.81%	99.99%	99.27%	90.27%
QF = 18	99.24%	99.02%	99.97%	100.00%	99.81%	91.80%
QF = 19	99.60%	99.17%	99.99%	100.00%	99.98%	92.12%
QF = 20	99.85%	99.68%	100.00%	99.99%	99.90%	96.87%
QF = 30	99.93%	99.94%	99.99%	99.98%	99.92%	99.88%
QF = 40	100.00%	98.81%	99.87%	100.00%	99.85%	93.30%
QF = 50	100.00%	99.22%	99.98%	100.00%	99.99%	93.52%
QF = 60	100.00%	99.72%	99.98%	100.00%	99.99%	96.86%
QF = 70	100.00%	99.55%	99.98%	100.00%	100.00%	95.23%
QF = 80	100.00%	99.71%	99.99%	100.00%	100.00%	95.29%
QF = 90	100.00%	99.86%	99.98%	100.00%	100.00%	98.45%
QF = 100	100.00%	99.86%	99.99%	100.00%	100.00%	98.45%

Table 13. General comparisons of runtime for encoding and decoding for high-resolution images.

Image	Lake	Bedroom	Castle	Dog	Ducks	Moon
Image size	8192 × 6144	4160 × 6240	4000 × 6000	3024 × 4032	2268 × 4032	9972 × 9972
Runtime for encoding (sec)						
DCT	3.4768	1.8335	1.7115	0.8803	0.6541	6.9974
Embedding	11.4198	6.3822	6.1445	2.0011	1.9945	22.7751
IDCT	10.0510	5.8818	5.6627	1.8034	1.8062	20.7060
Total runtime for encoding per pixel (μsec)						
	0.4957	0.5431	0.5633	0.3843	0.4872	0.5076
Runtime for decoding (sec)						
DCT	3.4710	2.0674	1.9587	0.6340	0.6430	7.3656
Extraction	11.0180	6.3239	5.9622	1.9266	1.9422	22.2101
Total runtime for decoding per pixel (μsec)						
	0.2879	0.3233	0.3300	0.2100	0.2827	0.2974

In Table 11, we explain the watermark composition and the capacity for embedding, as depicted in Section 3.3 and Table 1. We can determine the maximally allowable capacity and then decide the version of the QR code, the module size, and the number of shares in VSS. Taking *castle* as an instance, the maximally allowable capacity is $\frac{4000 \times 6000}{8 \times 8} = 375,000$ bits. Then, we choose QR code version 5 with 37×37 modules, and each module has 3×3 pixels. The binary QR code has resolution of 111×111 or 12,321 bits. For the VSS shares with 8-bit representation, we choose VSS(2, 3). All three shares should be embedded, implying $12,321 \cdot 3 \cdot 8 = 295,704$ bits for embedding. VSS(2, 3) denotes any two shares out of all three can reconstruct the QR code. The number '2' in VSS(2, 3) relates to VSS decoding. Only the number '3' in VSS(2, 3) is needed for the calculation of embedding capacity. Finally, we also provide the capacity ratio, which is the ratio of embedding capacity to the image resolution. In this way, the capacity ratio for *castle* is $\frac{295,704}{4000 \times 6000} = 0.01232$. Note that for *dog* and *ducks*, we choose VSS(1, 1) to imply that it is doable and implementable, and it does not provide threshold confidentiality. We only turn the binary QR code into the share, which is presented by 8 bits per pixel. Finally, the capacity ratios for all six images lie in a comparable range.

In Table 12, we depict the robustness of extracted watermarks under a wide range of JPEG attacks. With the proper selection of Δ in Table 10, for *lake*, *bedroom*, and *moon*, they can survive under QF = 15; for *castle*, *dog*, and *ducks*, they can survive under QF = 13. Performances on robustness are not image-dependent. With similar watermarked image quality and a similar capacity ratio, the capability for surviving JPEG attacks also lies within comparable QF values.

In Table 13, we provide the runtime for encoding and decoding. These are executed with CPU: 2700X and RAM: DDR4 32G. With our scheme, the encoder is composed of three parts: DCT, embedding, and IDCT. The encoding times of the three parts are provided. We can see that it takes longer to encode larger images. Due to the different sizes of the six images, for fair comparisons, the total runtime for encoding per pixel is also calculated. They lie within a comparable range.

With our scheme, the decoder is composed of two parts: DCT and extraction. The decoding times of the two parts are provided. We can see that it takes longer to encode larger images. Due to the different sizes of the six images, for fair comparisons, the total runtime for decoding per pixel is also calculated. They lie within a comparable range. It takes less time for decoding because only the extraction of the watermark is needed, and reconstruction of the image is not necessary. Hence, IDCT needs not be included for decoding.

With the simulations presented in Section 4, based on proposed implementation, our scheme can embed binary QR codes directly into conventional test images that have low resolutions and embed VSS shares into images taken by ourselves that have high resolutions. Six high-resolution images are applied to present the feasibility and practicality of our scheme. Performance metrics for robust watermarking, including imperceptibility, capacity, and robustness, are all considered. It is robust to JPEG attacks even when small QF values are encountered. Users may choose a large quantization stepsize to withstand extreme cases with extremely low QF values in JPEG attacks, with the sacrifice of degradation of watermarked image quality. The proposed scheme not only presents better performances over related schemes in the literature but also provides flexibility for users to choose the watermark corresponding to original images for authenticity and copyright protection.

5. Discussion

In addition to the simulation results, we would like to note the cross-disciplinary synthesis of our scheme with the following aspects.

Firstly, we discuss the theoretical breakthrough on resolution and capacity scaling. As explained in Section 3.3, for the $t \times t$ module in the version v QR code, the criterion $(1/v) \cdot t \leq 32$ should be satisfied. In the case of the male test image with a size of 1024×1024 , only a 1×1 module is achievable. When choosing the 2×2 module for the QR code, VSS becomes impractical for conventional test images. In the case of the lake image with size 8192×6144 , VSS shares can be calculated when the QR code is represented in 8-bit form, with luminance values ranging from 0 to 255.

Secondly, we would like to refer to the discovery of multi-layered error boundaries. Unlike DCT-based watermarking with QIM, where single-bit extraction errors degrade the watermark directly, our integrated framework discovers and defines a multi-layered error tolerance boundary. This boundary leverages a novel intersection of frequency-domain quantization, polynomial secret sharing thresholds for k out of n , and spatial majority voting mechanism. This architecture shifts the scientific paradigm from purely preventing bit errors to mathematically absorbing them, becoming nearly 100% functionally scannable, even under severe degradation. The three layers are described as follows.

1. Layer 1: QIM stepsize tuning of Δ . The framework allows the balance between imperceptibility and robustness using the quantization stepsize Δ . Traditional methods fail under low-quality factors, generally for QF values less than 20. Our framework proves that scaling Δ dynamically maps out a survival threshold, maintaining high BCR, with $\text{BCR} \geq 99.85\%$ down to a brutal QF = 1 stress test, as demonstrated in Table 9.
2. Layer 2: Cryptographic thresholding of $\text{VSS}(k, n)$. By converting the binary QR code into an 8-bit presentation prior to embedding, the framework utilizes Lagrange interpolation to introduce an explicit mathematical boundary. Even if individual shares experience severe adversarial attack conditions during transmission, the secret QR code is perfectly reconstructed with methods in Section 2.3 as long as any k out of n shares are successfully recovered.
3. Layer 3: Spatial module majority voting. To resolve extreme compression artifacts, such as blocking effects and loss of sharp edges, we implement a structural verification layer based on $t \times t$ pixel modules. For an odd-numbered matrix like a 3×3 module, if 5 or more pixels are correctly recovered, the entire module self-heals to its true binary state. For even-numbered matrices (e.g., 2×2), we mathematically model and assign the specific combinations to guarantee smooth reconstruction.

This multi-layered behavior represents a distinct information-theoretic contribution over standard single-layer blind bit extraction routines.

Thirdly, we would like to point out the difference between our method and previous methods of DCT-based watermarking with QIM. Previous methods optimized the imperceptibility and robustness of a single bitstream. Our framework tries to balance the quantization stepsize, not just against raw BCR but against the majority vote mechanics of spatially expanded VSS modules. By establishing custom quantization boundaries ranging from $\Delta = 1.9$ to $\Delta = 17.5$, our approach adaptively balances extreme image quality degradation to guarantee nearly 100% survivability, even under brutal JPEG compression attacks with very low QF values.

Next, for the scientific contribution of the VSS integration, it is a scientific contribution because it redefines the error threshold of a watermark. In standard watermarking, an error in a bit simply degrades the output linearly. In our QR-VSS scheme, the error propagation is governed by two layers: the polynomial threshold for k out of n shares,

and the physical layout of the QR finder patterns. Our approach discovers a new multi-layered robust boundary where a watermark can possess a high bit error rate natively but still achieve 100% functional reconstruction via the intersection of Lagrange interpolation and majority voting.

Moreover, we would also like to discuss the complexity analysis for memory requirements. With our scheme, the framework operates on a streaming or block-based architecture using non-overlapping 8×8 pixel blocks. Therefore, the system does not need to hold the entire high-resolution image structure in volatile memory for the core mathematical transforms.

The image is partitioned into blocks of size $8 \times 8 = 64$ pixels. The 2D-DCT and QIM calculations are executed locally within each block. The extra memory required to store the 64 localized intermediate luminance values $X(i, j)$ and their corresponding 64 DCT coefficients $Y(u, v)$ in Equation (4) is bounded by a constant, $O(64) = O(1)$.

The VSS module computes polynomial values pixel by pixel over a compact prime field of $\mathbb{Z}_{257}$. Consequently, the memory footprint required for generating or interpolating shares scales directly with the size of the QR watermark capacity. In the proposed framework, the total capacity of the QR watermark payload is strictly upper-bounded by the host image dimensions, specifically denoted as $\frac{M}{8} \times \frac{N}{8}$ blocks. To execute the polynomial evaluations, the system retains n coefficients in volatile memory. The theoretical memory consumption is, therefore, bounded by $\left(\frac{M}{8} \times \frac{N}{8} + n\right)$ bytes, supplemented by a minimal, fixed number of runtime ephemeral variables. Although rare boundary cases yielding a value of 256 in Equation (9) introduce negligible runtime overhead for positional tracking, this extra byte allocation is transient and can be safely omitted from the primary complexity estimation.

Since blocks are processed sequentially, the auxiliary memory footprint remains $O(1)$, or constant space, relative to the image size if pixels are streamed. The VSS memory consumption is limited compared to the image pixel data. If the entire image frame is loaded into memory as a pixel array for processing, the space complexity is $O(M \times N)$, scaling linearly with image resolution. For a 48 MP image, using standard 8-bit grayscale elements or 32-bit floating-point coefficients requires a highly manageable deterministic memory allocation, which leads to approximately 48 MB to 192 MB per color channel buffer.

In addition to performance metrics among imperceptibility, robustness, and capacity, we hope discussions in this section will bring a more comprehensive perspective to our framework.

6. Conclusions

In this study, we present a robust watermarking framework that integrates QR codes and visual secret sharing for practical applications. Images captured today, or images seen on the Internet, have dozens-of-times-higher resolutions than conventional test images. This expanded capacity makes the integration of visual secret sharing practical, adding an additional layer of cryptographic protection to the watermarking scheme. We choose the QR code for embedding into conventional images due to its limited capacity. We apply secret shares in VSS for embedding into images captured by ourselves due to the abundant capacity offered. The watermarking scheme can withstand JPEG compression attacks, and, hence, after the extraction of secret shares and calculations with VSS decoding, the reconstructed QR code can be scanned by users.

Simulation results validate the effectiveness of our algorithm and implementation of robust watermarking with visual secret sharing. Discussions may bring a comprehensive overview of the framework. Algorithm developers may look for the balance among

marked image quality, watermark capacity, and watermark robustness from an objective viewpoint. In addition, when we examine the reconstructed QR code after experiencing JPEG attacks in the marked image, it needs to be scannable and has the capability of linking to the designated URL. The proposed algorithm and implementation have achieved the goals mentioned above.

Author Contributions: Conceptualization, H.-C.H. and F.-C.C.; methodology, H.-C.H. and F.-C.C.; software, J.-E.L.; formal analysis, H.-C.H., F.-C.C., and J.-E.L.; writing—original draft preparation, H.-C.H., F.-C.C., and J.-E.L.; writing—review and editing, H.-C.H. and F.-C.C. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the National Science and Technology Council (Taiwan), under grant number MOST111-2221-E-390-015.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Data are contained in this article.

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

References

1. Yan, F.; Ji, X.; Dong, L.; Sun, W.; Li, Y. Universal and quality-preserving watermark removal based on unpaired learning. *IEEE Trans. Dependable Secur. Comput.* **2026**, *23*, 1767–1781. <https://doi.org/10.1109/TDSC.2025.3621004>.
2. You, J.; Zhou, Y. Two-stage watermark removal framework for spread spectrum watermarking. *IEEE Trans. Multimed.* **2024**, *26*, 7687–7699. <https://doi.org/10.1109/TMM.2024.3370380>.
3. Wu, S.; Lu, W.; Luo, X. Robust watermarking based on multi-layer watermark feature fusion. *IEEE Trans. Multimed.* **2025**, *27*, 5156–5169. <https://doi.org/10.1109/TMM.2025.3543079>.
4. Gao, G.; Wang, M.; Wu, B. Efficient robust reversible watermarking based on ZMs and integer wavelet transform. *IEEE Trans. Ind. Inform.* **2024**, *20*, 4115–4123. <https://doi.org/10.1109/TII.2023.3321101>.
5. Daniel, V.A.S.; Vijayalakshmi, K.; Pawar, P.P.; Kumar, D.; Bhuvanesh, A.; Christilda, A.J. Enhanced affinity propagation clustering with a modified extreme learning machine for segmentation and classification of hyperspectral imaging. *e-Prime-Adv. Electr. Eng. Electron. Energy* **2024**, *9*, 100704. <https://doi.org/10.1016/j.prime.2024.100704>.
6. Cedillo-Hernandez, M.; Cedillo-Hernandez, A.; Garcia-Ugalde, F.J.; Sanchez-Garcia, J.C. Semi-fragile watermarking scheme for high-resolution color images: Tamper identification, ownership authentication, and self-recovery. *Algorithms* **2026**, *19*, 28. <https://doi.org/10.3390/a19010028>.
7. Huang, H.C.; Chang, F.C.; Li, H.Y. Content-adaptive reversible data hiding with multi-stage prediction schemes. *Sensors* **2025**, *25*, 6228. <https://doi.org/10.3390/s25196228>.
8. Huang, H.C.; Fang, W.C. Authenticity preservation with histogram-based reversible data hiding and quadtree concepts. *Sensors* **2011**, *11*, 9717–9731. <https://doi.org/10.3390/s111009717>.
9. ISO/IEC 15415:2024; Information Technology: Automatic Identification and Data Capture Techniques—Bar Code Symbol Print Quality Test Specification—Two-Dimensional Symbols. ISO/IEC: Geneva, Switzerland, 2024; p. 46. Available online: <https://www.iso.org/standard/76876.html> (accessed on 13 July 2026).
10. Majnarić, I.; Jelkić, M.; Morić, M.; Hajdek, K. Print Quality Assessment of QR Code elements achieved by the digital thermal transfer process. *J. Imaging* **2026**, *12*, 86. <https://doi.org/10.3390/jimaging12020086>.
11. Chen, X.; Deng, J.; Chen, Y.; Li, C.; Fang, X.; Liu, C. Imprints: Mitigating watermark removal attacks with defensive watermarks. *IEEE Trans. Inf. Forensics Secur.* **2025**, *20*, 1866–1881. <https://doi.org/10.1109/TIFS.2025.3536299>.
12. SIPI Image Database-Misc. Available online: <https://sipi.usc.edu/database/database.php?volume=misc> (accessed on 13 July 2026).
13. Unsplash: Non-Copyrighted Images. Available online: <https://unsplash.com/images/stock/non-copyrighted> (accessed on 13 July 2026).
14. Yang, C.N.; Li, P.; Kuo, H.C. (k, n) secret image sharing scheme with privileged set. *J. Inf. Secur. Appl.* **2023**, *73*, 103413. <https://doi.org/10.1016/j.jisa.2022.103413>.

15. Kim, Y.; Jeong, O.; Moon, I.; Javidi, B. Multiparty random phase wrapping secret-sharing systems for visual data security. *IEEE Trans. Syst. Man Cybern. Syst.* **2025**, *55*, 3586–3600. <https://doi.org/10.1109/TSMC.2025.3541827>.
16. Adjero, D.A.; Lee, M.C. Scene-adaptive transform domain video partitioning. *IEEE Trans. Circuits Syst. Video Technol.* **2004**, *6*, 58–69. <https://doi.org/10.1109/TMM.2003.819578>.
17. Wang, Z.; Bovik, A.C.; Sheikh, H.R.; Simoncelli, E.P. Image quality assessment: From error visibility to structural similarity. *IEEE Trans. Image Process.* **2004**, *13*, 600–612. <https://doi.org/10.1109/TIP.2003.819861>.
18. Flynn, J.R.; Ward, S.; Abich, J.; Poole, D. Image quality assessment using the SSIM and the just noticeable difference paradigm. In *Proceedings of the 10th International Conference on Engineering Psychology and Cognitive Ergonomics, Las Vegas, NV, USA, 21–26 July 2013*; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2013; Volume 8019, pp. 23–30. https://doi.org/10.1007/978-3-642-39360-0_3.
19. Liu, Y.; Shi, C.; Hu, X.; Qi, C.; Niu, X. Robust and secure image watermarking algorithm resisting JPEG compression and cropping based on Reed Solomon (RS) codes. *Signal Image Video Process.* **2025**, *19*, 253. <https://doi.org/10.1007/s11760-024-03803-z>.
20. Ko, H.J.; Huang, C.T.; Horng, G.; Wang, S.J. Robust and blind image watermarking in DCT domain using inter-block coefficient correlation. *Inf. Sci.* **2020**, *517*, 128–147. <https://doi.org/10.1016/j.ins.2019.11.005>.
21. Artiles, J.A.P.; Chaves, D.P.B.; Pimentel, C. Robust image watermarking algorithm using chaotic sequences. *J. Inf. Secur. Appl.* **2022**, *68*, 103219. <https://doi.org/10.1016/j.jisa.2022.103219>.
22. Loan, N.A.; Hurrah, N.N.; Parah, S.A.; Lee, J.W.; Sheikh, J.A.; Bhat, G.M. Secure and robust digital image watermarking using coefficient differencing and chaotic encryption. *IEEE Access* **2018**, *6*, 19876–19897. <https://doi.org/10.1109/ACCESS.2018.2808172>.
23. Mao, J.; Tang, H.; Lyu, S.; Zhou, Z.; Cao, X. Content-aware quantization index modulation: Leveraging data statistics for enhanced image watermarking. *IEEE Trans. Inf. Forensics Secur.* **2023**, *19*, 1935–1947. <https://doi.org/10.1109/TIFS.2023.3342612>.
24. Chen, B.; Wornell, W.B. Quantization index modulation: A class of provably good methods for digital watermarking and information embedding. *IEEE Trans. Inform. Theory* **2001**, *47*, 1423–1443. <https://doi.org/10.1109/18.923725>.
25. Melman, A.; Evsutin, O. Hybrid metaheuristic algorithms for image watermarking: An experimental study. *Swarm Evol. Comput.* **2025**, *99*, 102163. <https://doi.org/10.1016/j.swevo.2025.102163>.
26. Melman, A.; Evsutin, O. Image watermarking based on a ratio of DCT coefficient sums using a gradient-based optimizer. *Comput. Electr. Eng.* **2024**, *117*, 109271. <https://doi.org/10.1016/j.compeleceng.2024.109271>.
27. Moosazadeh, M.; Ekbatanifard, G. A new DCT-based robust image watermarking method using teaching-learning-Based optimization. *J. Inf. Secur. Appl.* **2019**, *47*, 28–38. <https://doi.org/10.1016/j.jisa.2019.04.001>.
28. Chen, Y.H.; Huang, H.C. Coevolutionary genetic watermarking for owner identification. *Neural Comput. Appl.* **2015**, *26*, 291–298. <https://doi.org/10.1007/s00521-014-1615-z>.
29. Dolev, D.; Yao, A.C. On the security of public key protocols. *IEEE Trans. Inform. Theory* **1983**, *29*, 198–208. <https://doi.org/10.1109/tit.1983.1056650>.

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.