Chang et al. EURASIP Journal on Advances in Signal Processing (2015) 2015:11
DOI 10.1186/s13634-015-0196-z

EURASIP Journal on Advances in Signal Processing
a SpringerOpen Journal

RESEARCH Open Access

Progressive sharing of multiple images with sensitivity-controlled decoding

Sheng-Yu Chang, Suiang-Shyan Lee, Tzu-Min Yeh, Lee Shu-Teng Chen and Ja-Chen Lin*

Abstract

Secure sharing of digital images is becoming an important issue. Consequently, many schemes for ensuring image sharing security have been proposed. However, existing approaches focus on the sharing of a single image, rather than multiple images. We propose three kinds of sharing methods that progressively reveal *n* given secret images according to the sensitivity level of each image. Method 1 divides each secret image into *n* parts and then combines and hides the parts of the images to get *n* steganographic (stego) JPEG codes of equal importance. Method 2 is similar; however, it allocates different stego JPEG codes of different 'weights' to indicate their strength. Method 3 first applies traditional threshold-sharing to the *n* secret images, then progressively shares *k* keys, and finally combines the two sharing results to get *n* stego JPEG codes. In the recovery phase, various parameters are compared to a pre-specified low/middle/high (L/M/H) threshold and, according to the respective method, determine whether or not secret images are reconstructed and the quality of the images reconstructed. The results of experiments conducted verify the efficacy of our methods.

Keywords: Progressive sharing; Multiple images; Weighted sharing; Guardian stegos; Sensitivity-controlled decoding

1 Introduction

The Internet has become an integral part of human life and society. This public facility constantly transmitted both public and private information. Consequently, the protection of sensitive information transmitted through this medium has become an important issue. Blakley and Shamir [1,2] first conceptualized the idea of a (t, n) threshold secret sharing scheme, in which at least a minimum number t out of n participants are required in order to recover the secret. This scheme has been extended by various researchers [3-16] and successfully applied to activities such as protection of PDF files [12], visual cryptography [13,14], and network communication [15]. For digital media, many schemes for ensuring image sharing security have been proposed. For example, Thien and Lin [8] proposed using n shares, in which each share is t times smaller than the given secret image, to share a secret image. Wang and Shyu [4] proposed a scalable secret image sharing scheme. Lin and Tsai proposed image sharing schemes with authentication capabilities [9], or with reduction of share size [10]. Further, some approaches are devoted to progressively decoding secrets [3-7].

Besides sharing, the approaches using data hiding [17-19] or watermarking [20-24] have also offered other kinds of protection. In general, a hiding method can embed a secret file in a host image. In data hiding, the researchers usually consider the issues such as the size ratio between the secret file and the host image; and the impact on the host image due to embedding. As for the use of watermark, people can embed a watermark in a digital image in order to authenticate or claim the ownership of the digital image. In the design of watermarking methods, the researchers usually pay more attention to the work of resisting attacks such as copy attack, tampering, and cropping. Nowadays, the study of watermarks has covered not only software [21,24] but also hardware [22,23]. Notably, a sharing method often has a post-processing which utilizes data hiding or some kinds of authentication tool. This is because each generated share looks like noise and may attract the attention of hackers; whereas data hiding can hide the generated shares in ordinary images. An authentication tool might also be needed in order to verify the

* Correspondence: jclin@cs.nctu.edu.tw
Department of Computer Science, National Chiao Tung University, 1001 University Road, Hsinchu 30050, Taiwan

integrity. For example, ref. [12] uses the SHA-256 hash function to authenticate the file.

Our study here focuses on sharing. Among the existing image sharing approaches, the secret being shared is often assumed to be a single image, rather than multiple images. Repeated use of single-image sharing method often causes the user to neglect the cross relation between distinct images and makes the setting of recovery thresholds not quite suitable. For example, in the sharing process of the photos of criminals, if we only have single-image sharing software, we might just use one set of thresholds for all photos. However, if we have multiple-image sharing software, which requires us to input the security level of each photo being processed simultaneously, then we will be more likely to take a closer look of the case of each criminal, then distinguish the photos, and finally give a stricter threshold setting for the photos of the more serious crime offenders.

When multiple images are being shared, the fact that the security/sensitivity of some images might be higher than that of the other images has to be considered. In this paper, we consider how to share several secret images simultaneously. This paper proposes three progressive sharing methods (methods 1, 2, and 3) that use sensitivity-controlled decoding. The sensitive images, i.e., the secret images, are divided into several image groups according to security level, with the more sensitive groups requiring more steganographic images (stegos) to uncover the images they contain. Specifically, after sharing and hiding, all stegos in method 1 are of equal weight, whereas the stegos in method 2 are quite different; some have more weight, and hence their secret-hiding ability is more powerful than that of the other stegos. Finally, in method 3, some stegos are so powerful that they are called guardian stegos: in this method, no information can be revealed without a minimum number of guardian stegos. Thus, in our proposed methods, secret images in each security group are revealed progressively when a user receives enough stegos (method 1), the sum of the received weights is sufficient (method 2), or a sufficient number of guardian stegos is present (method 3).

2 Background and related work

2.1 Secret sharing: (*t*, *n*) sharing

Thien and Lin [8] proposed the (t, n) threshold method, which distributes a secret image among n shares. First, the secret image is divided into non-overlapping sectors of t pixels each. Then, the following polynomial is used to encode every sector:

$$f(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_{t-1} x^{t-1} (\bmod p), \quad (1)$$

where $a_0, \ldots, a_{t-1}$ are the t pixel values in a sector and x is a user-specified index. Here, p is a prime number (or p is a whole power of 2, such as 128 or 256, if the arithmetic +, −, ×, and ÷ are in terms of Galois field operations). Finally, a share, whose index is x, is generated after every encoded result of $f(x)$ is concatenated. Notably, n unique indices $\{x_1, x_2, \ldots, x_n\}$ are selected at the beginning in order to create n shares, and each share size is $1/t$ of the original secret image.

In the decoding phase, if at least a minimum number t of the n shares is available, the original secret image can be reconstructed using Lagrange interpolation. The secret image is revealed if at least a minimum number t of the n shares is gathered; otherwise, only noise is obtained.

2.2 Progressive sharing: [r_1&r_2&…&r_k; n] sharing

Chen and Lin [3] developed a progressive sharing method. They used k thresholds - specifically, $\{r_1 \le r_2 \le \ldots \le r_k\}$ - with each threshold less than or equal to n. For example, for [(2&3&4); n], the three threshold values are $r_1 = 2$, $r_2 = 3$, and $r_3 = 4$. Then, the image is partitioned into multiple sectors comprising nine ($= r_1 + r_2 + r_3$) pixels each. To share a sector - for example, the nine values {146, 167, 255, 60, 124, 165, 211, 73, 25} - first, the rearranging process illustrated in Figure 1 transforms the nine values into nine new values {230, 21, 159, 23, 83, 155, 227, 136, 207}. In the above, note that the binary representation of 230 is 11100110, exactly the first eight digits read from the first column in Figure 1. The first $r_1 = 2$ transformed set of values, {230, 21}, gives the first polynomial in Equation 2. The next $r_2 = 3$ transformed set of values, {159, 23, 83}, creates the second polynomial in Equation 3. The final $r_3 = 4$ transformed set of values, {155, 227, 136, 207}, creates the third polynomial in Equation 4.

$$f^{(1)}(x) = 230 + 21x (\bmod p) \quad (2)$$

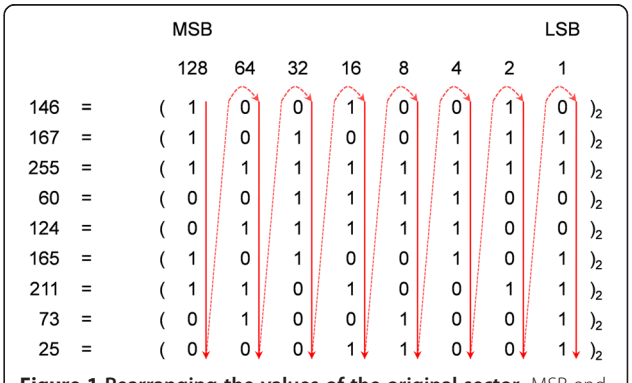

Figure 1 Rearranging the values of the original sector. MSB and LSB are the abbreviations for most and least significant bit, respectively.

$$f^{(2)}(x) = 159 + 23x + 83x^2 \,(\mathrm{mod}\, p) \quad (3)$$

$$f^{(3)}(x) = 155 + 227x + 136x^2 + 207x^3 \,(\mathrm{mod}\, p) \quad (4)$$

Here, as stated in Section 2.1, either let p be a prime number, or let p be a whole power of 2, such as 128 or 256 (if we do all arithmetic in the Galois field). Now, if any two of the generated shares are available, Lagrange interpolation can be used to reconstruct the two coefficients (230 and 21) in Equation 2. By reversing the rearranging process, we get the rough sector, {128, 128, 192, 0, 64, 128, 192, 64, 0}, of the original sector. If any three of the generated shares are available, we can reconstruct the 2 + 3 = 5 coefficients of Equations 2 and 3 and get an approximate sector, {144, 160, 248, 56, 112, 160, 208, 64, 16}, for the original sector. Finally, if any four of the generated shares are available, we can reconstruct the coefficients of Equations 2 to 4 and get the original sector, {146, 167, 255, 60, 124, 165, 211, 73, 25}, without errors.

3 Proposed methods

We propose three methods: method 1 is a basic progressive sharing method that divides n secret images into t groups according to sensitivity levels. In this method, for each j, the sensitivity level of the jth group must be lower than that of the $j + 1$th group. Further, the user provides several thresholds for each secret image group. For instance, if $r_1 \le r_2 \le \ldots \le r_k$ is given for a specified group, and if less than r_1 shadows are received, nothing can be displayed. However, if r_1 shadows are available, then the user can get a low-quality version of the images in that group. The more shadows obtained, the better the quality of the recovered images. Finally, if r_k shadows are available, then the user can recover the original images of that group without any errors. This paragraph just mentioned 'shadow'; and a shadow is formed of several 'shares'. In fact, in all three proposed methods, each shadow is formed of t shares (because each of the t groups offers a share to the mentioned shadow). The construction detail of the shadows will be in step 4 of the three encoding algorithms in Subsections 3.1.1, 3.2.1, and 3.3.1 below.

Method 2 assigns different weights to different 'cover' image groups. The smaller the weight value of a cover group, the smaller the number of shadows hidden in that cover group. The secret images are also partitioned into groups. For each secret image group, for example, secret group j, multiple threshold values (for instance $r_{j1} \le r_{j2} \le \ldots \le r_{jk}$) are specified by the user. Subsequently, during the decoding, if the sum of the weights of the received cover groups is at least r_{j1}, then the user can recover a low-quality version of the images in secret group j. The greater the sum of the received weights, the better the quality of the recovered secret images. Finally, if the sum of the weights equals r_{jk}, then we can recover all original images in secret group j without errors.

Method 3 designates some of the stego images to be guardian stegos. In this method, if a sufficient minimum number of these guardian stegos are received, then low-quality secret images can be reconstructed, as long as the number of received stego images is also at or above a minimum threshold value. The more guardian stegos received, the better the quality of the recovered images, as long as the number of received stego images is also at certain corresponding threshold values. Finally, if all the guardian stegos are received, then all the secret images can be reconstructed without errors, as long as the number of stego images received is also at or above certain threshold values.

3.1 Method 1: basic form (of sharing with sensitivity-controlled decoding)

3.1.1 Encoding phase

Input: n secret images $\{S_1, S_2, \ldots, S_n\}$, n cover images (each is in JPEG form), and t sets of 'type-r progressiveness thresholds', $\{[r_{11} \le r_{12} \le \ldots \le r_{1k}], [r_{21} \le r_{22} \le \ldots \le r_{2k}], \ldots, [r_{t1} \le r_{t2} \le \ldots \le r_{tk}]\}$.

Output: n JPEG stego codes.

Step 1: Divide $\{S_1, \ldots, S_n\}$ into t groups according to the sensitivity levels of $\{S_1, \ldots, S_n\}$. (For each $j = 1, \ldots, t - 1$, the sensitivity level of group j must be lower than that of group $j + 1$.)

Step 2: Rearrange the data sequence of each secret image as follows:

Step 2.1: For each non-overlapping 8×8 block, perform discrete cosine transform (DCT). Then, according to the zigzag order, only grab DCT values from the direct current (DC) term to the final non-zero value of the alternating current (AC) terms. (Notably, if a quantization of DCT coefficients has been used, then apply Hoffman coding to the residual image which is the difference image between the original image and the image decompressed from the quantized DCT coefficients.)

Step 2.2: For each DCT block, fill in zeroes so that the DCT value of the block is a multiple of RSUM, the local sum of the type-r progressiveness thresholds; i.e., $\mathrm{RSUM} = \mathrm{RSUM}_j = r_{j1} + r_{j2} + \ldots + r_{jk}$ if the image is in the jth group. Then, rearrange the data sequence of the DCT block in accordance with Figure 2.

Step 3: For each secret group $j = 1, 2, \ldots, t$, use $[(r_{j1} \& r_{j2} \& \ldots \& r_{jk}); n]$ progressive sharing to get n shares, which share the DCT data of each image in group j. (Remark: if lossless reconstruction is also wanted, then for each secret group $j = 1, 2, \ldots, t$, use r_{jk} as the threshold value in traditional (non-progressive) sharing to generate another n shares, which share the Huffman

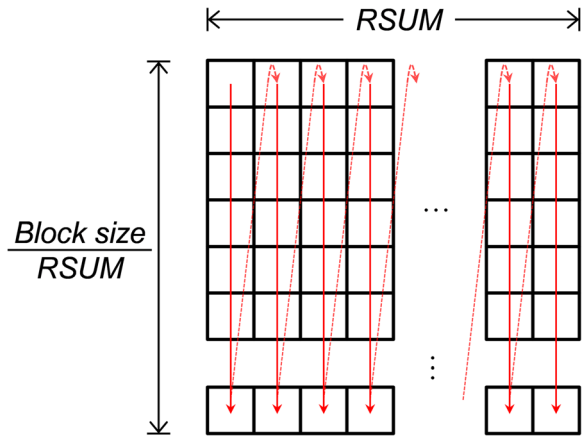

Figure 2 DCT value sequence rearrangement order. In Figure 2, the block is adjusted (by filling in zeroes) so that the number of DCT elements (in the block) is a multiple of the integer constant 'RSUM'.

codes (see step 2.1) of the residual images in group j. Now, for $i = 1$ to n, attach share i of Huffman code to share i of DCT data. This pairwise binding will reduce $n + n$ shares to n shares.)

Step 4: In step 3, each secret group generated n shares, namely, {ith share | $i = 1, 2, \ldots, n$}. Now, for $i = 1, 2, \ldots, n$, concatenate (i.e., physically link together) the ith shares across all t secret groups to get the ith shadow. Note that there are t secret groups, and each shadow receives one share from each secret group. Hence, each shadow is formed of t shares. For example, shadow 1 is in the form (share 1 of group 1, share 1 of group 2, ..., share 1 of group t).

Step 5: Use the JPEG data hiding method [17] to hide the n shadows in the respective n JPEG codes of the n cover images.

3.1.2 Decoding phase

If (any) r_{11} of the n stego images are available, then we can extract the shadows from the r_{11} stego images: they can then be used to reconstruct low-quality versions of all the secret images in group 1. If (any) r_{12} of the n stego images are available, the quality of the recovered group 1 secret images will be better. Finally, if (any) r_{1k} of the n stego images are available, then the recovered group 1 secret images will all be lossless. Similarly, for each $j = 2, \ldots, t$, if (any) $r_{j1}, r_{j2}, \ldots$ of the n stego images are available, we get the progressive recovery effect mentioned above for group j.

3.2 Method 2: sensitivity-controlled decoding using weights

3.2.1 Encoding phase

Input: n secret images $\{S_1, S_2, \ldots, S_n\}$, n cover images (each is in JPEG form), t sets of 'type-r progressiveness thresholds', $\{[r_{11} \le r_{12} \le \ldots \le r_{1k}], [r_{21} \le r_{22} \le \ldots \le r_{2k}], \ldots, [r_{t1} \le r_{t2} \le \ldots \le r_{tk}]\}$, and T positive integers $\{w_1, w_2, \ldots, w_T\}$ called 'weights'. Note: $w_1 + w_2 + \ldots + w_T = n$.

Output: n JPEG stego codes.

Steps 1 to 4: Do steps 1 to 4 in Section 3.1.1.

Step 5: Assign the n cover images to T cover groups so that each cover group has at least one cover image. Then, for each $j = 1, 2, \ldots, T$, assign weight w_j to cover group j.

Step 6: Use the JPEG data hiding method [17] to hide the w_1 shadows in the JPEGs of the first cover group, the w_2 shadows in the JPEGs of the second cover group, and so on. Since $w_1 + w_2 + \ldots + w_T = n$, hiding of the n generated shadows is complete when the final w_T shadows are hidden in the tth cover group.

3.2.2 Decoding phase

The decoding is carried out according to the total sum of the weights of the received cover groups. If the total sum of the received weights corresponds to r_{11}, then we can extract the r_{11} shadows from the received cover groups and reconstruct a low-quality version of all the images in secret group 1. If the total sum of the received weights corresponds to r_{12}, then the recovered images of secret group 1 will be of a better quality. Finally, if the total sum of the received weights corresponds to r_{1k}, then the recovered images of secret group 1 will be lossless. Analogously, for each $j = 2, \ldots, t$, if the total sum of the received weights correspond to $r_{j1}, r_{j2}, \ldots$, or r_{jk}, we get the above progressive recovery effect for the jth secret group.

3.3 Method 3: sensitivity-controlled decoding with guardian stegos

Thus far, for each secret image group, both methods 1 and 2 used 'multiple' progressiveness thresholds to control the progressive effect of that secret image group (for example, the parameters $[r_{11} \le r_{12} \le \ldots \le r_{1k}]$ are used for secret image group 1, the parameters $[r_{21} \le r_{22} \le \ldots \le r_{2k}]$ are used for secret image group 2, and so on). In contrast, method 3 uses only one r_j as the 'single' threshold for the jth secret image group (true for each $j = 1, 2, \ldots$). The progressive effect of method 3 is achieved by other types of parameters (parameters of type q, rather than of type r).

3.3.1 Encoding phase

Input: n secret images $\{S_1, S_2, \ldots, S_n\}$, n cover images (each is in JPEG form), t positive integer parameters, $\{r_1 \le r_2 \le \ldots \le r_t\}$, k keys, $\{Key_1, Key_2, \ldots, Key_k\}$, and k positive integers, $[q_1 \le q_2 \le \ldots \le q_k = k]$, called 'type-$q$ progressiveness parameters'. (Note: type-q parameters are for the progressive sharing of keys, which is different

from the t sets of type-r thresholds of methods 1 and 2; methods 1 and 2 use no keys.)

Output: n JPEG stego codes.

Step 1: Do step 1 in Section 3.1.1.

Step 2: Rearrange the data sequence of each secret image and encrypt each value as follows:

Step 2.1: Do step 2.1 in Section 3.1.1.

Step 2.2: Partition the DCT coefficients of each block into k non-overlapping regions according to the zigzag sequence. Region 1 is the most important because it corresponds to the lowest-frequency area, followed by region 2, and so on. Then, for each $i = 1, 2, \ldots, k$, use Key_i to encrypt the DCT values belonging to region i. Finally, use Key_k again to encrypt the Huffman code generated in step 2.1.

Step 3: For each $j = 1, 2, \ldots, t$, use r_j as the threshold value in the threshold-sharing to create n shares that share the encrypted data sequence of each image of the jth secret group.

Step 4: For $i = 1, 2, \ldots, n$, combine the ith shares of all the secret images in the input to get the ith shadow.

Step 5: For each $i = 1, 2, \ldots, k$, use q_i as the threshold value in the (q_i, k) threshold-sharing to share Key_i among k key-shares. (consequently, among these k key-shares of Key_i, any q_i key-shares can recover Key_i without errors.)

Step 6: For $i = 1, 2, \ldots, k$, combine the ith key-shares of all keys in the database to get the ith key-shadow.

Step 7: Use the JPEG data hiding method [17] to hide the n shadows in the respective n JPEG codes of the n cover images.

Step 8: Choose k of the n cover images and use the JPEG data hiding method [17] to hide their respective k key-shadows in the k JPEG codes of the k cover images chosen. Note: the k stego images generated in this way are called 'guardian stegos'.

Thus, for k keys, there are k guardian stegos. Further, the actual number of progressive levels is less than or equal to k (equal to k if all k progressiveness parameters, $[q_1 \le q_2 \le \ldots \le q_k]$, are mutually distinct, i.e., $q_1 < q_2 < \ldots < q_k$).

3.3.2 Decoding phase

If (any) r_1 of the n stego images are available, we can extract the q_1 key-shadows and the r_1 shadows from the r_1 stego images, as long as the r_1 stego images include q_1 guardian stegos. Subsequently, we can recover the encrypted version of all the secret images in secret group 1, reconstruct the key Key_1, and use Key_1 to decrypt the encrypted version of the images in secret group 1. This process reveals the low-quality version of all the secret images in secret group 1. If the r_1 stego images include q_2 guardian stegos, we can reconstruct the key Key_2, resulting in the recovered version of the secret images in group 1 having improved quality. Finally, if the r_1 stego images include k guardian stegos, the recovered secret images in group 1 are all lossless. Similarly, for each $j = 2, \ldots, t$, if (any) r_j of the n stego images are available, then, as long as the r_j stego images include the $q_1, q_2, \ldots,$ or q_k guardian stegos, we get the above progressive recovery effect for the secret images in group j.

4 Experimental results

We conducted experiments 1, 2, and 3 for methods 1, 2, and 3, respectively. We utilized the six 512×512 cover images, {Barbara, Lake, Couple, Baboon, Indian, Bridge}, shown in Figure 3 in all the experiments. We also utilized the six secret images, {House, Cameraman, Lena, Pepper, Jet, Blonde}, shown in Figure 4 in each experiment; however, because of the limitations imposed on size by the different methods, the width and height of each secret image were smaller in experiments 1 and 2, and larger in experiment 3.

We measured the quality of each stego image and recovered image using PSNR, defined as

$$\mathrm{PSNR} = 10 \times log_{10} \frac{255^2}{\mathrm{MSE}} \tag{5}$$

Here, the mean square error (MSE) is given by

$$\mathrm{MSE} = \frac{1}{\mathrm{height} \times \mathrm{width}} \sum_{i=1}^{\mathrm{height}} \sum_{j=1}^{\mathrm{width}} \left(\mathrm{pixel}_{ij} - \mathrm{pixel}'_{ij}\right)^2 \tag{6}$$

for an image with height × width pixels, and pixel_{ij} and pixel'_{ij} are, respectively, the value of the pixel at position (i, j) of the two compared images. For the readers' convenience, structural similarity [25] (SSIM) is also listed. Notably, the better the image quality, the closer the distance between SSIM value and 1.

Figure 3 The six 512×512 cover images: {Barbara, Lake, Couple, Baboon, Indian, Bridge}.

Figure 4 The six secret images: {House, Cameraman, Lena, Pepper, Jet, Blonde}.

4.1 Experimental results for method 1

In this experiment, the inputs comprised the six 128 × 128 secret images and the six 512 × 512 cover images (Figure 3). We divided the six images into three groups according to the sensitivity levels of the six secret images, [House, Cameraman], [Lena, Pepper], and [Jet, Blonde], and used $[(r_{11}\&r_{12}\&r_{13});\ n] = [(2\&3\&4);\ 6]$ in the progressive sharing to distribute the first group's images, [House, Cameraman], among {share$_1$ … share$_6$}. Similarly, we used $[(r_{21}\&r_{22}\&r_{23});\ n] = [(3\&4\&5);\ 6]$ in the progressive sharing to distribute the second group's images, [Lena, Pepper], among {share$_1$ … share$_6$}. Finally, we used $[(r_{31}\&r_{32}\&r_{33});\ n] = [(4\&5\&6);\ 6]$ in the progressive sharing to distribute the third group's images, [Jet, Blonde], among {share$_1$ … share$_6$}.

To complete the encoding, we constructed the first shadow by integrating all share$_1$s, the second shadow by integrating all share$_2$s, and so on. Finally, we used the JPEG data hiding method [17] to hide the six shadows in the respective six JPEG codes of the six cover images.

In the decoding phase, for the scenario where two of the six stego images were available, we first extracted the two shadows hidden in the two available stego images. Then, by inverse-sharing and because $r_{11} = 2$, we were able to recover the low-quality version of both secret images, [House, Cameraman], in group 1. For the scenario where three of the six stego images were available, we first extracted the three shadows hidden in the three available stego images. Then, by inverse-sharing and because $r_{21} = 3$ and $r_{12} = 3$, we were able to recover the low-quality version of both secret images, [Lena, Pepper], in group 2, and the medium-quality version of both secret images, [House, Cameraman], in group 1, respectively.

Similarly, for the scenario where any four of the six stego images were available, because $r_{31} = 4$, $r_{22} = 4$, and $r_{13} = 4$, we were able to recover the low-quality version of both secret images, [Jet, Blonde], in group 3, the medium-quality version of both secret images, [Lena, Pepper], in group 2, and the lossless version of both secret images, [House, Cameraman], in group 1, respectively. For the scenario where any five of the six stego images were available, because $r_{32} = 5$, $r_{13} = 4 < 5$, and $r_{23} = 5$, we were able to recover the medium-quality version of both secret images, [Jet, Blonde], in group 3, and the lossless version of each secret image in the first group, [Lena, Pepper], and the second group, [House, Cameraman], respectively. Finally, for the scenario where all six stego images were available, because $r_{j3} \le 6$ for each $j = 1, 2, 3$, we were able to recover all six secret images without error, irrespective of the group to which they belonged.

Table 1 shows the quality of the progressively recovered secret images during the decoding phase. Note that, after encoding, when we decompressed the six JPEG stego codes, which contained the secrets hiding in them, the PSNRs of the decompressed images were between 39.6 and 42 dB, as shown in Table 1. The quality of the images revealed on level 1 (i.e., the low-quality version) is between 24.95 and 27.33 dB, and the quality revealed on level 2 (i.e., medium-quality version) is between 30.35 and 33.09 dB. The secret images were recovered without errors on level 3 of the reconstruction.

4.2 Experimental results for method 2

In this experiment, the input comprised the six 128 × 128 secret images, the six 512 × 512 cover images (Figure 3), and three weight values {1, 2, 3}. The six images were again divided into three groups according to the sensitivity levels of the secret images: [House, Cameraman], [Lena, Pepper], and [Jet, Blonde]. Then, in the progressive sharing, $[(r_{11}\&r_{12});\ n] = [(3\&4);\ 6]$ was used to distribute each of the first group's secret images, [House, Cameraman], among {share$_1$ … share$_6$} so that the 'rough' recovery of any image (say, House) in this group would need $r_{11} = 3$ shares, whereas the lossless recovery of that image would need $r_{12} = 4$ shares. Similarly, we used $[(r_{21}\&r_{22});\ n] = [(4\&5);\ 6]$ in the progressive sharing to distribute each of the second group's images, [Lena, Pepper], among {share$_1$ … share$_6$}. Finally, we used $[(r_{31}\&r_{32});\ n] = [(5\&6);\ 6]$ in the progressive sharing to distribute each of the third group's images, [Jet, Blonde], among {share$_1$ … share$_6$}.

The first shadow was generated by integrating the share$_1$s of all six secret images, the second by integrating the share$_2$s of all six secret images, and so on. Let |SS| denote the size of a shadow. We also partitioned the six cover images into three groups, [Barbara, Lake], [Couple, Baboon], and [Indian, Bridge], and assigned them weights 1, 2, and 3, respectively. Finally, we bound all the JPEG codes of the cover images of the first cover group together as a unit. Then, we treated this unit as a

Table 1 Image quality of method 1 (three-level progressive sharing)

Secret images	Progressive thresholds (r_{j1}, r_{j2}, r_{j3})	Image quality of recovery on level 1			Image quality of recovery on level 2			Quality of stego images			
		PSNR	SSIM	MSE	PSNR	SSIM	MSE	Stego images	PSNR	SSIM	MSE
House	2&3&4	26.29	0.795	152.8	33.06	0.905	32.1	Barbara	39.60	0.975	7.1
Cameraman	2&3&4	24.95	0.787	208.0	30.35	0.897	60.0	Lake	42.00	0.977	4.1
Lena	3&4&5	26.40	0.777	149.0	32.15	0.902	39.6	Couple	41.63	0.972	4.5
Pepper	3&4&5	26.05	0.804	161.5	33.09	0.924	31.9	Baboon	41.22	0.979	4.9
Jet	4&5&6	25.86	0.817	168.7	31.66	0.914	44.4	Indian	41.92	0.971	4.2
Blonde	4&5&6	27.33	0.797	120.2	32.83	0.905	33.9	Bridge	40.50	0.977	5.8

On level 3, the recovery is lossless.

cover medium and used the JPEG data hiding method [17] to hide only one shadow in this cover medium (we hid only one shadow because $w_1 = 1$). The shadow hidden here was shadow #1, with size being $w_1 \times |SS| = |SS|$. Then, we bound all the JPEG codes of the cover images of the second cover group together as a unit and used the hiding method [17] to hide two ($2 = w_2$) shadows (i.e., shadows #2 and #3) in this unit; thus, the secret data hidden in the second cover group had size $w_2 \times |SS| = 2 \times |SS|$. Finally, we bound all the JPEG codes of the cover images of the third cover group together as a unit and used the hiding method to hide in this unit three ($3 = w_3$) shadows (i.e., shadows #4, #5, and #6). Hence, the data being hidden in the third cover group had size $w_3 \times |SS| = 3 \times |SS|$.

For the scenario where we received all stego JPEG codes of the first cover group, [Barbara, Lake], we extracted the only shadow (i.e., shadow #1) hidden in the first cover group. However, nothing could be displayed because the weight, $w_1 = 1$, was too small. When we did not receive the first cover group, but instead, received all the stego JPEG codes of the second cover group, [Couple, Baboon], we were only able to extract the two shadows (i.e., shadows #2 and #3) hidden in the second cover group. Similarly, nothing could be displayed because the weights $w_2 = 2$ were still not sufficiently large. Finally, for a scenario where we received neither the first cover group nor the second cover group but received all the stego JPEG codes of the third cover group, [Indian, Bridge], we first extracted the three shadows (i.e., shadows #4, #5, and #6) hidden in the third cover group. Then, by inverse-sharing and because threshold $r_{11} = 3$, we were able to recover the low-quality version of each secret image in the first secret group, [House, Cameraman].

For the scenario where we received all four stego JPEG codes for both the first cover group, [Barbara, Lake], and the second cover group, [Couple, Baboon], we first extracted the $w_1 = 1$ shadow hidden in the first cover group and then extracted the $w_2 = 2$ shadows hidden in the second cover group. Thus, we extracted $w_1 + w_2 = 1 + 2 = 3$ shadows, namely, {shadows #1, #2, and #3}. Then, by inverse-sharing and because $r_{11} = 3$, we were able to recover the low-quality version of each secret image in the first secret group, [House, Cameraman].

Similarly, for the scenario where we received all four stego JPEG codes of both the first cover group, [Barbara, Lake], and the third cover group, [Indian, Bridge], we were able to extract $w_1 + w_3 = 1 + 3 = 4$ shadows, namely, {shadows #1, #4, #5, and #6}. Thus, because $r_{21} = 4$ and $r_{12} = 4$, we were able to recover the low-quality version of each secret image in the second secret group, [Lena, Pepper], and the lossless version of each secret image in the first secret group, [House, Cameraman], respectively.

For the scenario where we received all four stego JPEG codes of both the second cover group, [Couple, Baboon], and the third cover group, [Indian, Bridge], we were able to extract $w_2 + w_3 = 2 + 3 = 5$ shadows, namely, {shadows #2, #3, #4, #5, and #6}. Thus, because $r_{31} = 5$, $r_{12} = 4 < 5$, and $r_{22} = 5$, we were able to recover the low-quality version of each secret image in the third secret group, and the lossless version of each secret image in the first and second secret groups, respectively. Finally, for the scenario where we received all six stego JPEG codes, because $r_{j2} \le 6$ for each $j = 1, 2, 3$, we were able to recover all six secret images without errors, irrespective of the secret group to which they belonged.

Table 2 shows the quality of the progressively recovered secret images during the decoding phase. Note that, in the encoding phase, when we decompressed the six JPEG stego codes, which contained the secrets hiding in them, the PSNRs of the decompressed images were between 39.26 and 44.17 dB. The revealed secret images' quality on level 1 (i.e., low-quality version) of the reconstruction was between 28.21 dB and 30.71 dB. All secret images on level 2 of the reconstruction were recovered without errors.

4.3 Experimental results for method 3

Here, the input comprised the six 232×232 secret images, the six 512×512 cover images (Figure 3), three positive integer parameters $\{4 \le 5 \le 6\}$ for secret image

Table 2 Image quality of method 2 (two-level progressive sharing)

Secret images	Progressive thresholds (r_{j1}&r_{j2})	Image quality of recovery on level 1			Image quality of recovery on level 2	Quality of stego images			
		PSNR	SSIM	MSE		Stego images	PSNR	SSIM	MSE
House	3&4	30.71	0.883	55.17	Lossless	Barbara	43.75	0.980	2.73
Cameraman	3&4	28.21	0.866	98.18	Lossless	Lake	44.17	0.979	2.48
Lena	4&5	29.63	0.871	70.80	Lossless	Couple	42.10	0.978	4.00
Pepper	4&5	29.97	0.891	65.43	Lossless	Baboon	42.31	0.985	3.81
Jet	5&6	28.55	0.883	90.70	Lossless	Indian	41.29	0.975	4.82
Blonde	5&6	30.33	0.870	60.18	Lossless	Bridge	39.26	0.981	7.69

sharing (the values 4, 5, and 6 are for image groups 1, 2, and 3, respectively), three keys for encryption, and three integers $\{q_1 = 2, q_2 = 2, \text{and } q_3 = 3\}$ called type-q progressiveness parameters for the sharing of 'keys'.

We again divided the six images into three groups according to the sensitivity levels of the six secret images: secret group 1, lowest sensitivity, comprised [House]; secret group 2, moderate sensitivity, comprised [Cameraman, Lena]; and secret group 3, highest sensitivity, comprised [Pepper, Jet, Blonde]. Then, we encrypted each secret image using all three keys.

We then used $(r_1, n) = (4, 6)$ in secret sharing to share the first secret group, i.e., to share the encrypted House, among $\{\text{share}_1 \ldots \text{share}_6\}$. Similarly, we used $(r_2, n) = (5, 6)$ in secret sharing to share the second secret group (i.e., the encrypted Cameraman and the encrypted Lena) among $\{\text{share}_1 \ldots \text{share}_6\}$. Finally, we used $(r_3, n) = (6, 6)$ in secret sharing to share the third group's encrypted secret images [Pepper, Jet, Blonde] among $\{\text{share}_1 \ldots \text{share}_6\}$. The first shadow was generated by integrating the share_1s of all six secret images, the second by integrating the share_2s of all six secret images, and so on.

The thresholds to share the three keys $\{Key_1, Key_2, Key_3\}$ were, respectively, $q_1 = 2$, $q_2 = 2$, and $q_3 = 3$. Hence, for $i = 1, 2, 3$, we used $(q_i, 3)$ sharing to share the numerical value Key_i among $k = 3$ key-shares, so that any q_i of the $k = 3$ generated key-shares (of Key_i) could recover Key_i. Then, for $i = 1, 2, 3$, we combined the ith key-shares of all three keys to get the ith key-shadow.

Next, we used the JPEG data hiding method [17] to hide the six image-shadows in the respective six JPEG codes of the six cover images. Finally, we chose $k = 3$ of the six cover images and hid the $k = 3$ key-shadows in the $k = 3$ JPEG codes of the chosen $k = 3$ cover images; for example, {Barbara, Lake, Couple}. These three stego images were designated the 'guardian stegos'.

With any two of the three guardian stegos, we were able to first extract the two key-shadows hidden in the two available guardian stegos. Then, by the inverse progressive sharing process, we were able to recover Key_1 and Key_2 because their thresholds were $q_1 = 2$, and $q_2 = 2$, respectively. Because two of the three guardian stegos were already available, when any two of the three *non*-guardian stegos were also available, we had $2 + 2 = 4$ shares. We first extracted the four image-shadows hidden in the four available stego images (i.e., two guardian stegos and two *non*-guardian stegos). Then, by inverse-sharing and decryption and because $r_1 = 4$, we were able to recover the low-quality version of the secret image in the first group. Although we only had two keys (instead of three keys), we were still able to decrypt the first several (low-quality) DCT Coefficients (see step 2 in Section 3.3.1). This is why we can decrypt low-quality versions of an image even when not all three keys are available.

Let us now consider another scenario. We assumed that two of the three guardian stegos were already available; hence, $\{Key_1 \text{ and } Key_2\}$ were known. Consequently, because all $6 - 3 = 3$ *non*-guardian stegos were available, we first extracted the $2 + 3 = 5$ image-shadows hidden in the five available stego images. Then, by inverse-sharing and decryption, the low-quality version of each secret image in the second group were recovered because the threshold for the second image group was assumed to be $r_2 = 5$. However, since the total number of stego images received was only five, the secret images in the third group still could not be recovered because the threshold for the third image group was assumed to be $r_3 = 6$.

For the scenario where all three guardian stegos were available, we first extracted the three key-shadows hidden in the three guardian stegos. Then, by inverse-sharing, we were able to recover all three keys because the largest threshold for the keys was assumed to be $q_3 = 3$ when we earlier distributed the three keys among the three key-shadows. Since all three guardian stegos were now available, if any one of the $6 - 3 = 3$ *non*-guardian stegos was also available, then, since all three keys were already extracted, we were able to recover the lossless version of the secret image in the first group because $3 + 1 = 4$ and the threshold for image group 1 was assumed to be $r_1 = 4$. In the case where (any) two of the three *non*-guardian stegos were available, we were able to recover the

lossless version of each secret image in the second group because 3 + 2 = 5 and the threshold for image group 2 was assumed to be $r_2 = 5$. Finally, for the scenario where all three non-guardian stegos were available, we were able to recover the lossless version of each secret image in the third group because 3 + 3 = 6 and the threshold for image in group 3 was assumed to be $r_3 = 6$. The experimental results are listed in Table 3. Note that the thresholds for the sharing of the three keys are $\{q_1 = q_2 = 2$, and $q_3 = 3\}$; thus, there are only two levels to control the recovery of the keys; namely, the collection of two guardian stegos versus the collection of three guardian stegos. Thus, to view secret images, the effective number of progressive levels is also only two.

Note also that, if only one of the three guardian stegos is available, then no secret image can be recovered, even if all three *non*-guardian stegos are available. This is because the three encryption keys are shared and hidden in the guardian stegos, and the smallest threshold $q_1 = min\{q_1, q_2, q_3\}$ to recover at least one key was already set to $q_1 = 2$.

5 Discussion and comparison

5.1 Summary and discussion

Our proposed method 1 is a progressive sharing sensitivity-controlled decoding method; i.e., the decoding is conducted according to the sensitivity level of each image. Images with the same sensitivity level constitute a group. Each secret image in an image group is shared among n shares, and the shares of all images are properly combined to get n shadows with equal significance; consequently, there is no need to worry about which shadow is lost or transmitted first. The n shadows are hidden in the JPEG codes of n cover images to get n stego JPEG codes. If the number of received stegos corresponds to the lowest threshold of an image group, then the rough version of each secret image in that group can be revealed. The higher the number of stegos received, the better the quality of the recovered secret images. In particular, when the number of stego images received corresponds to the highest threshold (considering all thresholds for all groups), then all secret images in all groups can be recovered without errors.

Our proposed method 2 is also a progressive sharing sensitivity-controlled decoding method; however, it differs from method 1 in that 'weights' are used in method 2. We divide the n 'cover' images into several groups and equip each cover group with a weight specially assigned to that group. Then, according to the weight of each cover group, we hide some of the n secret shadows in the cover group.

Subsequently, decoding is conducted according to the total sum of the weights of the received cover groups. If the sum of the received weights corresponds to the lowest threshold of a secret group, then all secret images of that secret group can be recovered with a low quality. The larger the sum of the received weights, the better the quality of the recovered secret images. Finally, if the sum of the received weights corresponds to the highest threshold of a secret group, then the recovered secret images of that secret group are lossless.

Both progressive methods (methods 1 and 2) increase the shadow size after using multiple thresholds. Therefore, in our proposed method 3, we use a different technique to progressively share multiple secret images. In method 3, if the number of received guardian stegos corresponds to the lowest threshold, as long as the number of received stego images also corresponds to the threshold value of a secret image group, then the rough version of each secret image in that secret group can be revealed. The more guardian images received, the better the quality of the recovered secret images, as long as the number of received stego images also corresponds to certain threshold values. In particular, when the number of received guardian stegos corresponds to the highest threshold value, then all secret images can be recovered without errors, as long as the number of received stego images also corresponds to certain threshold values.

Compared with methods 1 and 2, method 3 has a tighter restriction in the recovery phase: nothing can be

Table 3 Image quality of method 3 (the secret images are recovered in two levels)

Secret images	Threshold r to recover the secret image	Image quality on low-level recovery			Image quality on high-level recovery	Image quality of stego			
		PSNR	SSIM	MSE		Stego images	PSNR	SSIM	MSE
House	4	31.53	0.858	45.7	Lossless	Barbara	40.07	0.979	6.40
Cameraman	5	27.11	0.853	126.5	Lossless	Lake	41.57	0.978	4.53
Lena	5	29.38	0.869	75.0	Lossless	Couple	41.04	0.978	5.12
Pepper	6*	No such level*			Lossless	Baboon	41.41	0.980	4.70
Jet	6*	No such level*			Lossless	Indian	41.56	0.976	4.54
Blonde	6*	No such level*			Lossless	Bridge	40.08	0.978	6.38

*Since we used $(r, n) = (6, 6)$ in the secret sharing of the third group's secret images, [Pepper, Jet, Blonde], these three images cannot be viewed unless all six stego images have been collected. Consequently, only high-level PSNR, i.e., only the lossless version, exists.

displayed without a sufficient number of guardian stegos. Therefore, methods 1 and 2 are more suitable for a public company whose owners are (public) stock holders. The more shadows (stocks) or the more weights appear in the meeting, the more secret details can be unveiled. In contrast, method 3 is more suitable for a family-owned private company in which all the decision-making must first get the permission of the persons in charge, or at least, get the majority agreement of the committee board.

In Table 4, we list the advantages and disadvantages of the three proposed methods. Notably, about the issue of stability, method 1 is the most stable one, as explained below. In method 2, the recovered versions of secret images are identical to that of method 1. However, stego images' quality is less stable in method 2, for the stegos' quality is influenced by the matching between the weights $\{w_i\}$ and the hiding capacity of the cover groups $\{CG_i\}$. When one of the weights is particularly large, the instability becomes obvious. For example, if $\{w_1 = 1, w_2 = 1, w_3 = 4\}$ and if the three cover groups have similar hiding capacity, then distinct stego groups might have very distinct qualities. In Table 2, where $\{w_1 = 1, w_2 = 2, w_3 = 3\}$, the quality of the image Bridge, which is in stego group 3, is also worse than the quality of the {Barbara, Lake} in stego group 1. Finally, method 3 is also less stable than method 1 because some assignment to the values of the type-r parameter might cut the effective number of progressive levels, as will be seen in Table 5 and a paragraph near the end of Section 5.5.

Now we analyze the precision of the recovered secret images. Since all three methods can produce error-free recovery as the highest-quality recovery, we focus our comparison on the lowest-quality version, i.e., the

Table 4 A comparison between the three proposed methods

Methods	Characteristic	Suitable environment	Advantage	Disadvantage
Method 1	a) Basic form of our progressive sharing/viewing.	All participants, i.e., all holders of stegos, must be of equal importance.	a) Simple and stable.	Shadow size is larger than that of method 3.
	b) Every stego image is of the same significance.		b) Every stego image hides the same amount of secrets. Therefore, there is no need to worry about which cover image should hide more.	
Method 2	a) The recovered versions of the secret images are identical to that of method 1.	The owners of some stegos are more important than the owners of other stegos.	a) Method 1 is just a special case of method 2.	a) Shadow size is larger than that of method 3.
	b) However, the stego images have different weights in method 2.		b) Hence, compared to method 1, method 2 has more possible ways to control the unveiling of secret images.	b) The hiding capacity of some covers might be insufficient (or a severe impact on some cover images might exist), if a weight value is much larger than other weight values.
				c) The stego images' quality is less stable than that of methods 1 and 3.
				d) Therefore, a careful matching between weights and covers might be needed. (In general, assign larger weights to the cover groups of larger size.)
Method 3	a) Using the so-called guardian stegos.	a) Suitable for a company which is controlled by a committee (the committee cannot allow any unveiling of secrets without the approval of a certain percentage of the committee members).	a) Some stegos are guardian stegos, and they form the committee to guard the disclosure of secrets (the unveiling of secrets cannot happen if many guardian stegos disapprove it).	a) The social rank of non-guardian stegos is very low. If the number of received guardian stegos is less than the minimal threshold value, then the secret images has no chance to be unveiled (even if 'every' non-guardian stego's holder wants to unveil the secret images).
	b) Keys are used in method 3.	b) Also suitable for the protection of images which are very sensitive.	b) The committee has the absolute rights to turn down the disclosure of a secret.	b) Some values of parameter r make the number of progressive levels reduced from the assigned value to a smaller value.
			c) Smaller shadow size.	
			d) Better security than methods 1 and 2.	

Table 5 Image quality of method 3 (the secret images are recovered in three levels)

Secret images	The (*r*, *n*) in sharing	Low-level recovery		Moderate-level recovery		High-level recovery	Stego images	Quality of stegos	
		PSNR	MSE	PSNR	MSE			PSNR	MSE
House	(4, 6)	29.89	66.69	35.72	17.42	Lossless	Barbara	40.14	6.30
Cameraman	(5, 6)	N/A	N/A	29.19	78.36	Lossless	Lake	41.56	4.54
Lena	(5, 6)	N/A	N/A	31.22	49.10	Lossless	Couple	41.16	4.98
Pepper	(6, 6)	N/A	N/A	N/A	N/A	Lossless	Baboon	41.29	4.83
Jet	(6, 6)	N/A	N/A	N/A	N/A	Lossless	Indian	41.47	4.64
Blonde	(6, 6)	N/A	N/A	N/A	N/A	Lossless	Bridge	40.17	6.25

N/A, no such level exists.

recovery on level 1. Methods 1 and 2 give identical recovered versions of secret images, so we only need to compare method 1 to method 3. In method 1, as analyzed in Section 5.5, when $(r_{j1} \le r_{j2} \le \ldots \le r_{jk})$ are utilized as the k progressive thresholds to share an image in secret group j, the lowest version's quality is determined by the ratio $r_{j1}/(r_{j1} + r_{j2} + \ldots + r_{jk})$. The larger the ratio value, the better the precision. Therefore, The best level-1 quality occurs when $k = 2$ and $r_{j1}/(r_{j1} + r_{j2} + \ldots + r_{jk}) = r_{j1}/(r_{j1} + r_{j2})$ is almost 1/2. In this case, as analyzed in Section 5.5, about $r_{j1}/(r_{j1} + r_{j2}) = 50\%$ of the rearranged DCT data are utilized to recover level 1 version. On the other hand, for method 3, if q_1 of the k guardian stegos are available, then the lowest version's quality is determined by the ratio area(region 1) ÷ [area(region 1) + … + area(region k)], where {region 1, … region k} are the k non-overlapping regions that partitioned the DCT coefficients in step 2.2 of Section 3.3.1. Since we had the freedom to assign any percentage of the DCT data to region 1, this area-ratio can be as low as 1%, or as high as 99%. Now, compared to the 50% of method 1, we can say that the lowest-quality recovery of method 3 can be either worse or better than the lowest-quality recovery of method 1. The precision comparison between the methods is thus case-by-case and inconclusive.

5.2 Comparison with reported methods

Our methods are progressive sharing methods. Functionality comparisons between our methods and various other progressive sharing schemes are shown in Table 6. Our methods' decoding is according to the sensitivity levels of different secret groups. In Table 6, all other schemes consider one secret image instead of multiple secret images. Furthermore, note that, in our method 2, distinct groups of 'cover' images are also assigned distinct weights.

The shadow size of method 1 is equal to that of method 2; method 3 has the smallest shadow size. As shown in Table 7, the shadow size is small in each of our three methods; thus, the shadow can be easily hidden in the JPEG codes of cover images. The sizes associated with the various methods are given below. In the traditional (t, n) secret sharing method, the size of the shadow is only $1/t$ of the original secret data. In our proposed methods 1 and 2, when we use $[(r_1\&r_2\&\ldots\&r_k); n]$ progressive sharing method to share some secret data, the size of each shadow is $k/(r_1 + r_2 + \ldots + r_k)$ times smaller than that of the original secret data, as explained below. We process $r_1 + r_2 + \ldots + r_k$ values together each time. The first r_1 values are shared by (r_1, n) sharing; thus, each shadow receives one value after sharing these r_1 values. Similarly, the next r_2 values are shared by (r_2, n) sharing; thus, each shadow receives one value after sharing these r_2 values, and so on. Therefore, when we consider the sharing of these $r_1 + r_2 + \ldots + r_k$ values; it is obvious that each shadow receives $1 + 1 + \ldots + 1 = k$ values generated from the sharing of these $r_1 + r_2 + \ldots + r_k$ values. As a result, the size of each shadow is $k/(r_1 + r_2 + \ldots + r_k)$ of the original size of the secret. Note that k is the number of progressiveness thresholds, $\{r_1 \ldots r_k\}$, being used. Therefore, if the maximal threshold r_k of a progressive sharing, $[(r_1\&r_2\&\ldots\&r_k); n]$, is equal to the single threshold t of non-progressive sharing, then both progressive scheme and non-progressive scheme can recover the original data without errors if t shares are received. However, the inequality

$$\begin{aligned} & k/(r_1 + r_2 + \ldots + r_{k-1} + r_k) \\ & \quad = k/(r_1 + r_2 + \ldots + r_{k-1} + t) \\ & \quad > (k/(t + t + \ldots + t)) = k/kt = 1/t \end{aligned} \tag{7}$$

tells us that the shadow size generated by progressive sharing is larger than the shadow size generated by non-progressive sharing. This is the price of being progressive. For instance, comparing a (4, n) non-progressive share and a [(3&4); n] progressive share, it can be seen that both schemes can recover secrets without errors when four shadows are received. However, if only three shadows are received, then the progressive scheme can still recover a 'rough' version, whereas the non-progressive scheme cannot. The shadow size generated by (4, n) non-progressive sharing is $S/4 = 0.25S$ (assuming that S is the

Table 6 Functionality comparisons between various proposed progressive sharing schemes and our sharing scheme

Schemes	Cover images format	Multiple secret images	Different sensitivity levels for different secret image groups	Different weights for different cover image groups	Use of guardian stegos
Chen and Lin [3]	Uncompressed cover images	No	No	No	No
Wang and Shyu [4]	Uncompressed cover images	No	No*	No	No
Hung et al. [5]	Uncompressed cover images	No	No	No	No
Chen and Lin [6]	Compressed cover images	No	No	No	No
Our scheme	Compressed cover images	Yes	Yes	Method 2	Method 3

*Wang and Shyu [4] used different sensitivity levels for different areas of an image.

size of the secret file); whereas the shadow size generated by [(3&4); n] progressive sharing is $2S/(3+4) = 0.286S$. If the number of progressive levels increases, for example, from a two-level scheme [(3&4); n] to a three-level scheme [(2&3&4); n], then the shadow size also increases and becomes $3S/(2+3+4) = 0.333S$.

Table 7 compares the shadow sizes of various progressive sharing schemes. Assume that the given secret image is the 512×512 grayscale image Lena, and the progressiveness thresholds are [(3&4&5&6), 6] for 'all' schemes. In Table 7, since our proposed methods 1 and 2 are designed for multiple secrets, we let the first secret group have only one secret image (Lena), and all other secret groups be empty (have no secret image). Then, we use [(r_{11}&r_{12}&r_{13}&r_{14}), n] = [(3&4&5&6), 6] as the (local) progressiveness thresholds for the secret group containing Lena. In Table 7, it can be seen that among all lossless methods, our scheme has the smallest shadow size. Our shadow size is even smaller than that of the lossy method proposed by Hung et. al. [5]. Using a small shadow size is important in every sharing method. If the shadow is small, the n shadows can be transmitted quickly, storage space can be saved, and shadows can be hidden easily in other media.

Table 8 summarizes the PSNRs of the recovered images and stego images, when three progressive levels were used in each reported progressive sharing scheme. Our three methods consider multiple images, so we have a PSNR range, rather than a single PSNR value. Each reported method has its own setting of parameters and is too tedious to list, so almost all experimental values in Table 8 were quoted directly from the cited papers. From Table 8, we can see that, like most of other single-secret progressive methods, all three multiple-secret methods of ours can also achieve lossless recovery of secret images; however, our impact to host images is smallest because our stego images have highest PSNR values.

5.3 Parameters' values

5.3.1 Parameters of methods 1 and 2

- k (the number of progressive levels)

We suggest the use of $k = 2$ or $k = 3$. Use $k = 2$ if the user wishes that the recovery of each secret image has two levels (low-quality vs. lossless). Use $k = 3$ if the user wishes that the recovery of each secret image has three levels (low-quality, medium-quality, and lossless). The value of k should not be large, because each 8-by-8 DCT block only have 64 coefficients, and many of them are zeros. For example, using $k = 8$ is impractical, for $\{r_{j1}, r_{j2}, \ldots, r_{j8}\} = \{2, 3, 4, 5, 6, 7, 8, 9\}$ implies that every RSUM $= 2+3+4+5+6+7+8+9 = 44$ DCT coefficients are bound together and processed as a unit in the progressive sharing process, but the 64 DCT coefficients might not have so many non-zeros. As a result,

Table 7 Comparison of shadow sizes among various progressive sharing schemes

Scheme	Size of each shadow (smaller is better) (bytes)	Quality of the best reconstructed versions (dB)	Shadow size ÷ (512 × 512) (%)
Chen and Lin [3]	58,254	Lossless	22.22
Wang and Shyu [4]	131,072	Lossless	50.00
Hung et al. [5]	30,720	37.04	11.72
Chen and Lin [6]	33,802	Lossless	12.89
*Our methods 1 and 2	28,023	Lossless	10.69

*Our method 3 is even more economic (less than 28,023 byes).

Table 8 Comparison of PSNRs and MSEs among various progressive sharing schemes (three levels)

Scheme	PSNR (MSE) of recovery on level 1	PSNR (MSE) of recovery on level 2	PSNR (MSE) of recovery on level 3	PSNR (MSE) of stego
Chen and Lin [3] (three experiments for 'Lena', with various parameter settings)	13.15 to 31.11 (MSE = 50.4 to 3148.3)	20.00 to 46.38 (MSE = 1.50 to 650.2)	Lossless	34.62 to 36.43 (MSE = 14.8 to 22.4)
Wang and Shyu [4]	*N/A	*N/A	Lossless	*N/A
Hung et al. [5] (Lena)	28.00 (MSE = 103.1)	32.67 (MSE = 35.1)	37.04 (MSE = 12.9)	35.68 to 35.86 (MSE = 16.9 to 17.6)
Chen and Lin [6] (lossy version)	28.37 (MSE = 94.7)	34.12 (MSE = 25.18)	39.79 (MSE = 6.8)	32.97 to 38.96 (MSE = 8.3 to 32.8)
Our method 1	24.50 to 28.23 (MSE = 97.7 to 230.7)	30.26 to 34.85 (MSE = 21.3 to 61.2)	Lossless	39.60 to 42.00 (MSE = 4.1 to 7.2)
Our method 2	24.50 to 28.23 (MSE = 97.7 to 230.7)	30.26 to 34.85 (MSE = 21.3 to 61.2)	Lossless	37.21 to 43.23 (MSE = 3.1 to 12.4)
Our method 3	26.92 to 30.53 (MSE = 57.6 to 132.2)	29.19 to 35.72 (MSE = 17.4 to 78.4)	Lossless	40.74 to 42.35 (MSE = 3.8 to 5.5)

*Wang and Shyu [4] purposely let some area of the image be blank in their low-level recovery. Thus, PSNR is not available.

either the method cannot be implemented or a waste of the shadow space is introduced in the sharing process.

- r_{j1}

For each secret image group, for example, secret group j, both methods 1 and 2 utilize k progressive threshold values ($2 \le r_{j1} \le r_{j2} \le \ldots \le r_{jk} \le n$). Here, r_{j1} means that, people cannot recover any version of the images in secret group j unless at least r_{j1} stegos (in method 1) are received, or unless the sum of the weights of the received cover groups is at least r_{j1} (in method 2). Therefore, we suggest the use of a large value for r_{j1} (for example, $r_{j1} = \lceil n/2 \rceil$ or $\lceil 2n/3 \rceil$) if the images in secret group j are very sensitive; otherwise, just use a small value for r_{j1} (for example, $r_{j1} = 2$ or 3). Note that $r_{j1} = \lceil n/2 \rceil$ means at least one half of the shadows must be available before the lowest-quality version of the sensitive images can be reconstructed.

- r_{jk}

Since r_{jk} is related to the lossless recovery of the images in secret group j, let r_{jk} be a very large value (for example, n or $n-1$) if secret group j is very sensitive. Notably, using $r_{jk} = n$ means that all cover images must be received before the lossless version of secret group j can be recovered. Using large value for the parameter r_{jk} can avoid the stealing of the lossless version of the images in secret group j; however, it also weakens the missing-allowable benefit of the sharing approach. Therefore, do not let r_{jk} be as large as n or $n-1$, unless the images in secret group j are very sensitive.

- The weights $\{w_1, w_2, \ldots, w_T\}$ and the cover groups (CG)

Method 2 uses T weights $\{w_1, w_2, \ldots, w_T\}$ with $w_1 + w_2 + \ldots + w_T = n$. The n given cover images are assigned to T cover groups, and cover group j (CG_j) hides w_j shadows, true for each $j = 1, 2, \ldots, T$. Therefore, $|CG_j|$, which denotes the number of cover images in CG_j, cannot be small if w_j is large. To explain this, without the loss of generality, assume that there are, again, $n = 6$ secret images and $n = 6$ cover images. If $T = 3$ and the weights of the three cover groups are $w_1 = 1$, $w_2 = 1$, and $w_3 = 4$, respectively, then the number of cover images in the three cover groups can be {1, 1, 4} or {1, 2, 3} or {2, 2, 2}. Among them, the combination {2, 2, 2} is the worst, as explained below. Since $\{|CG_j|\}_{j=1,2,3} = \{2, 2, 2\}$, one shadow ($w_1 = 1$) is hidden in the two cover images of CG_1; one shadow ($w_2 = 1$) is hidden in the two cover images of CG_2; but four shadows ($w_3 = 4$) are hidden in the two cover images of CG_3. The impact on the two cover images of CG_3 will be too large due to the fact that each cover image must hide $(w_3/|CG_3|) = (4/2) = 2$ shadows. Therefore, to avoid that the quality of some stego images become too low, we suggest that the value of $|CG_j|$ cannot be small if w_j is large.

In the last paragraph, when $\{w_1 = 1, w_2 = 1, w_3 = 4\}$, although the partition $\{|CG_j| = w_j\} = \{1, 1, 4\}$ of the six cover images will not cause big impact on the cover images, the fact that $|CG_3| = 4$ means that these four cover images are bound together, and the stego file size of cover group 3 is very large (four times larger than that of cover group 1). This might give the manager of group 3 some inconvenience. As a compromise between the

stego quality and convenience, $\{|CG_j|\} = \{1, 2, 3\}$ might be the best tradeoff when $\{w_1 = 1, w_2 = 1, w_3 = 4\}$.

The other reason why we do not use $\{|CG_j| = w_j\}_{j=1, 2, \ldots, T} = \{|CG_j| = w_j\}_{j=1,\ldots,3} = \{1, 1, 4\}$ is as explained below. Using $|CG_j| = w_j$ for any j will make method 2 very similar to method 1: each cover image hide exactly one secret image. Hence, the quality of both the stegos (and the recovered secret images) is identical between methods 1 and 2. The only difference is that the stegos in method 1 is not bound together to form stego groups. So, from the viewpoint of recovery, if $|CG_j| = w_j \; \forall j = 1, 2, \ldots, T$, then method 1 is more convenient than method 2. In method 1, the possible number of shadows for decoding can be any number from 1 through n, because the number of received shadows equals to the number of received stegos. In method 2, however, only $w_1 = 1$, $w_2 = 1$, $w_3 = 4$, $w_1 + w_2 = 2$, $w_1 + w_3 = 5$, $w_1 + w_2 + w_3 = 6$ shadows are possible combination for the number of received shadows. In this case, the recovery using three shadows will never happen. Therefore, if 3 is one of the progressive threshold values ($2 \le r_{j1} \le r_{j2} \le \ldots \le r_{jk} \le 6$) for some cover group j, then each image in that group will not have k-levels progressive decoding, provided that method 2 is used and $\{|CG_1| = 1 = w_1;\ |CG_2| = 1 = w_2;\ |CG_3| = 4 = w_3\}$.

5.3.2 Parameters of method 3

- The threshold r_j for the sharing of secret group j ($j = 1, 2, \ldots, t$)

The basic requirement is $r_j \in \{2, \ldots, n\}$. Assign a very large value (for example, n or $n - 1$) to r_j, if secret group j is very sensitive. In general, using a larger r_j value can make it harder to steal the images in secret group j; however, this also reduces the missing-allowable benefit of the sharing approach: the corruption of two stegos will make the recovery of secret group j become almost impossible. Therefore, do not let r_j be as large as n or $n - 1$, unless secret group j is very sensitive. On the other hand, use $r_j = 2$ or $r_j = 3$ if secret group j is of very low sensitivity.

- The k progressiveness parameters $[q_1 \le q_2 \le \ldots \le q_k = k]$ of type-q

For each $j = 1, 2, \ldots, t$, if (any) r_j of the n stego images are available, then, since $r_1 \le r_2 \le \ldots \le r_j$, we can reconstruct the 'encrypted' version of each secret image in {secret group 1, …, secret group j}. After that, there are k possible levels of decryption. Assume that the r_j received stego images include q_m guardian stegos. Then, we can recover the m keys $\{Key_1, \ldots, Key_m\}$; and hence, reconstruct m of the k regions of the DCT coefficients. Consequently, each image in secret groups {1, …, j} can be revealed using the mth-level recovery quality.

As for the ith progressiveness parameter q_i in the set $[q_1 \le q_2 \le \ldots \le q_k = k]$, just assign a large value (such as k or $k - 1$) to q_i if we wish that the ith-level recovery should not be done without the joint attendance of many guardian stegos.

Below we give some examples to illustrate how the values of the k progressiveness parameters $[q_1 \le q_2 \le \ldots \le q_k = k]$ affect the decoding results. Assume that we have received r_j stego images; so, the 'encrypted' version of each secret image in secret groups 1, 2, …, j has been known. Then, e.g., 1, let $[q_1 = q_2 = \ldots = q_{k-1} = 2$, and $q_k = k]$. If the r_j received stego images include two or more guardian stegos, then we can recover almost every region of DCT coefficients. Consequently, each image in secret groups {1, …, j} can be revealed with a very good quality; e.g., 2, let $[q_1 = q_2 = \ldots = q_{k-1} = k - 1$, and $q_k = k]$. If the r_j received stego images include $k - 2$ of all guardian stegos, then we still cannot decrypt any region of any secret image; e.g., 3, let $[q_1 = q_2 = 2$; and $q_i = i$ for $i = 3, 4, \ldots, k]$. If the r_j received stego images include two guardian stegos, then we can recover each image in secret groups {1, …, j} with level 1 quality. If three guardian stegos are included, then the recovery quality is better, i.e., of level 2. If four guardian stegos are included, then the recovery quality is of level 3, and so on. Therefore, this is a progressive effect with many levels.

- The sharing thresholds $\{r_j\}$ vs. the progressiveness parameters $[q_1 \le q_2 \le \ldots \le q_k = k]$

The influence by r_j is local. For example, $r_j = 3$ only means that at least three out of n stegos are required in order to get the encrypted version of secret group j. On the contrary, the influence of q_i is global (across all secret groups), because $q_i = 3$ means that at least three of the k guardian stegos must be available in order to decrypt region i of the DCT coefficients of all secret images in all secret groups.

- k

Notably, in step 8 of the algorithm, we chose k of the n cover images to hide the respective k key-shadows to get the k guardian stegos. Hence, $k < n$ is a natural requirement ($k = n$ makes every stego become a guardian stego; but this is a system not quite meaningful, for it is just like a company in which every employee is in the supervisor committee of the company). On the other hand, $k = 1$ makes no progressive effect (because the k-level-decryption reduces to one-level-decryption). Thus, $k \in \{2, \ldots, n - 1\}$. For example, in our experiments, since there were $n = 6$ images, so k is chosen from {2, 3, 4, 5}.

Of course, the larger the value of k, the more the number of recovery levels. Notably, k also affects the image quality recovered on the lowest level, as explained below. In our algorithm, the DCT coefficients are partitioned into k regions. If the partition is uniform among regions, then smaller value of k means that each region has more DCT coefficients; therefore, the level 1 reconstruction of secret images will have better quality.

5.4 Steganography and security issues

The protection of images in our system is with several check points: a) the stranger must extract the shadows which we hid in the JPEG stegos; b) the stranger must intercept enough number of shadows (if he knows which stegos to intercept); c) we can send or store stegos using distinct channels or computers, and our decoding allows that some channels or computers are destroyed, for the missing-allowable property of threshold-sharing; d) the interceptors must intercept sufficient number of stegos before they can try to obtain sensitive images, for the decoding using insufficient number of shadows are extremely difficult, as analyzed later in this subsection; e) some of our methods are with multiple keys, and thus increase the difficulty of hackers.

The reasons that we use the JPEG data hiding method [17] to embed our n shadows in n JPEG codes are as follows: 1) Compared to spatial-domain stegos, JPEG code can save storage space and maybe also reduce the chance of attracting attackers; 2) JPEG compression disturbs the correlation between adjacent pixels of an image, so the permutation pre-processing employed in certain image sharing schemes [4,8,10] before sharing can be omitted. 3) As for the security of our JPEG codes, the size of the JPEG code (without hiding any secret), with the quality factors being in the range 10 to 95, is between 8, 119, and 94,581 bytes for many gray-level images. The size of our JPEG stego-codes listed in Tables 1, 2, and 3 are all in this range. Therefore, the attackers will not be suspicious about the size of our JPEG stego codes. 4) Note that the JPEG hiding method [17] has been shown to resist Chi-square [26] and StegDetect [27] attacks, reducing the chance that the attackers notice the existence of our shadows in the JPEG stego codes. 5) To summarize, the hiding in the JPEG stego-code is less notable.

Below we analyze the probability to get the sensitive image through 'guessing' when the number of received shadows is less than the minimal requirement. In methods 1 and 2, for each image in secret group j, let the $[(r_{j1}\&r_{j2}\&\ldots\&r_{jk}); n]$ progressive sharing be utilized to get n shares, which share the DCT data of the image. Without the loss of generality, let the image be 128×128, and $[(r_{j1}\&r_{j2}\&\ldots\&r_{jk}); n] = [(3\&4); 6]$. Therefore, RSUM = 3 + 4 = 7, and the image has $(128 \times 128)/(8 \times 8) = 16 \times 16 = 256$ DCT blocks. In our experiments, some images have about 21 of the 64 DCT coefficients are non-zeros on the average. So, in the rearranged DCT data, each block has about 21 numbers. The first $r_{j1}/\mathrm{RSUM} = 3/(3 + 4) = 3/7$ of the 21 numbers are shared using $(3, n)$ sharing, and the next $r_{j2}/\mathrm{RSUM} = 4/(3 + 4) = 4/7$ of the 21 numbers are shared using $(4, n)$ sharing. In the decoding, if a person does not receive at least r_{j1} shadows; for example, assume that he only receives $r_{j1} - 1 = 3 - 1 = 2$ shadows; then for a three-coefficient polynomial like $f(x) = a_0 + a_1x + a_2x^2 = 109 + 23x + 83x^2$, although he knows $f(1)$ and $f(2)$, he still cannot know the three coefficients are (109, 23, 83). The only thing he knows is a table, i.e., if $a_1 = 0$, then $(a_1, a_2) = \ldots$; if $a_0 = 1$, then $(a_1, a_2) = \ldots$; if $a_0 = 2$, then $(a_1, a_2) = \ldots$; and so on. Now, if each number is in the range 0 to 255, then the probability to get correct $(a_0, a_1, a_2) = (109, 23, 83)$ is 1/256. Since the first 9 of the 21 numbers use $r_{j1} = 3$ as the threshold value, the chance that the stranger gets these 9 values from the two shadows that he owns is $(1/256)^{9/3} = (1/256)^3$. As for the next $21 - 9 = 12$ numbers, since they are shared using $r_{j2} = 4$ as the threshold value, the chance that the stranger gets these 12 values from $r_{j2} - 1 = 4 - 1 = 3$ shadows is $(1/256)^{12/4} = (1/256)^3$. However, for the 12 numbers, which are shared using $r_{j2} = 4$ as the threshold value, the probability that the stranger gets these 12 values from two shadows is much less than $(1/256)^{12/4} = (1/256)^3$. For example, for a four-coefficient polynomial like $g(x) = b_0 + b_1x + b_2x^2 + b_3x^3 = 78 + 43x + 65x^2 + 114x^3$, although the stranger knows two values such as $g(1)$ and $g(2)$ from the two shadows, he still cannot know the four coefficients are (78, 43, 65, 114). The only thing he knows is a two-dimensional table like 'if $(b_0, b_1) = (0, 0)$, then $(b_2, b_3) = \ldots$.; if $(b_0, b_1) = (0, 1)$, then $(b_2, b_3) = \ldots..$;' Consequently, if each number is in the range 0 to 255, then the probability to get correct $(b_0, b_1, b_2, b_3) = (78, 43, 65, 114)$ is $(1/256)^2$. Hence, the change to get these 12 values from two shadows is $[(1/256)^2]^{12/4} = (1/256)^6$. Together, the change to get the $9 + 12 = 21$ values of the block from two shadows is $(1/256)^3 \times (1/256)^6 = (1/256)^9$. For an image of w-by-h pixels, there are $(w \times h)/(8 \times 8)$ blocks. So the chance to get the rearranged DCT coefficients of the image is $[(1/256)^9]^{(w\times h)/(8\times 8)}$. If the sensitive image is 128×128, then the chance is $[(1/256)^9]^{16\times 16} = (1/256)^{9\times 256} = (256)^{-2,304} = 10^{-5,548}$. If the sensitive image is 512×512, then the chance is $(1/256)^{9\times 4,096} = (256)^{-36,864} = 10^{-88,777}$.

As for method 3, for each image in secret group j, method 3 uses (r_j, n) sharing to create n shares which share the 'encrypted' DCT data of the image. Without the loss of generality, still let the image be 128×128 and the sharing threshold be $r_j = 3$. Therefore, the image has $(128 \times 128)/(8 \times 8) = 16 \times 16 = 256$ DCT blocks. Still assume that the image has about 21 of the 64 DCT

coefficients are non-zeros on the average. Then, from the above analysis, in the decoding, if a person only receives $r_{j1} - 1 = 3 - 1 = 2$ shadows, then the chance that the stranger gets these 21 encrypted values of the block from the two shadows is $(1/256)^{21/3} = (1/256)^7$. For an image of w-by-h pixels, there are $(w \times h)/(8 \times 8)$ blocks. So the chance to get the encrypted DCT coefficients of the image is $[(1/256)^7]^{(wh/64)}$. If the sensitive image is 128-by-128, then the chance is $[(1/256)^7]^{16\times16} = (1/256)^{7\times256} = (256)^{-1,792} = 10^{-4,315}$. If the sensitive image is 512-by-512, then the chance is $(1/256)^{7\times4,096} = (256)^{-28,672} = 10^{-69,049}$.

Notably, with this very small chance, what the stranger gets using the two shadows is only the 'encrypted' DCT coefficients. He still needs to guess a) the number of keys, b) the value of each key (if the number of guardian stegos that he owns is below a required threshold value), c) the encryption method that we used (k distinct region can use k distinct encryption methods), d) the way we partitioned the non-zero into k regions, and so on.

5.5 Variation of parameters affects the recovery results of secret images

In the experiment of Section 4.1, the image House was progressively shared using $(r_{11}\&r_{12}\&r_{13}) = (2\&3\&4)$. The current section discusses how the variation of parameters' values affect the recovery results. Hence, let the new $(r_{11}\&r_{12}\&r_{13})$ be (3&4&5) and (4&5&6), respectively. As shown in Table 9, the reconstruction quality is improved, i.e., $MSE_{(4\&5\&6)} < MSE_{(3\&4\&5)} < MSE_{(2\&3\&4)}$, no matter in level 1's reconstruction or in level 2's. In fact, even if we replace the image House by other secret images, we still have $MSE_{(4\&5\&6)} < MSE_{(3\&4\&5)} < MSE_{(2\&3\&4)}$. Likewise, as shown in Table 10, in the two-level experiments, we also observe that $MSE_{(5\&6)} < MSE_{(4\&5)} < MSE_{(3\&4)} < MSE_{(2\&3)}$. This statement is still true when Lena or Pepper is replaced by other images. The reason is explained below. In our design for methods 1 and 2, the rearranged DCT data are shared. In fact, these data are partitioned into $RSUM_j = r_{j1} + r_{j2} + \ldots + r_{jk}$ parts. The first r_{j1} parts are shared using $(r_{j1}; n)$ sharing, the next r_{j2} parts are shared using $(r_{j2}; n)$ sharing, and so on. In the recovery on level 1, i.e., when r_{j1} of the n shadows are available, we can recover about $r_{j1}/(r_{j1} + r_{j2} + \ldots + r_{jk})$ of the rearranged DCT data. In the recovery on level 2, i.e., when r_{j2} of the n shadows are available, we can recover about $(r_{j1} + r_{j2})/(r_{j1} + r_{j2} + \ldots + r_{jk})$ of the rearranged DCT data. And so on. For example, in Table 9, when $(r_{j1}, r_{j2}, r_{j3}) = (2, 3, 4)$, the recovery on level 1 can recover about $r_{j1}/(r_{j1} + r_{j2} + \ldots + r_{jk}) = 2/(2 + 3 + 4) = 2/9$ of the rearranged DCT data. The recovery on level 2 can recover about $(r_{j1} + r_{j2})/(r_{j1} + r_{j2} + \ldots + r_{jk}) = (2 + 3)/(2 + 3 + 4) = 5/9$ of the rearranged DCT data. Analogously, when $(r_{j1}, r_{j2}, r_{j3}) = (3, 4, 5)$, the recovery on level 1 can recover about

Table 9 Quality of the recovery for methods 1 and 2 (three levels)

Secret images	Progressive thresholds (r_{j1}, r_{j2}, r_{j3})	Image quality of the recovery on level 1			Image quality of the recovery on level 2			Image quality of the recovery on level 3
		PSNR	SSIM	MSE	PSNR	SSIM	MSE	
House	2&3&4	26.29	0.806	152.75	33.06	0.909	32.07	Lossless
	3&4&5	26.84	0.816	134.42	34.06	0.915	25.47	Lossless
	4&5&6	28.23	0.844	97.53	34.85	0.917	21.27	Lossless
Cameraman	2&3&4	24.95	0.786	207.66	30.35	0.894	59.98	Lossless
	3&4&5	25.58	0.805	179.78	31.00	0.901	51.65	Lossless
	4&5&6	26.05	0.818	161.20	31.53	0.904	45.64	Lossless
Lena	2&3&4	25.50	0.738	182.87	31.25	0.895	48.72	Lossless
	3&4&5	26.40	0.784	148.76	32.15	0.906	39.55	Lossless
	4&5&6	27.39	0.815	118.55	32.95	0.911	32.96	Lossless
Pepper	2&3&4	25.14	0.764	199.03	32.04	0.916	40.64	Lossless
	3&4&5	26.05	0.800	161.45	33.09	0.923	31.85	Lossless
	4&5&6	27.04	0.824	128.52	33.43	0.924	29.50	Lossless
Jet	2&3&4	24.50	0.772	230.19	30.26	0.905	61.16	Lossless
	3&4&5	25.29	0.794	191.95	31.08	0.913	50.60	Lossless
	4&5&6	25.86	0.819	168.33	31.66	0.914	44.26	Lossless
Blonde	2&3&4	25.92	0.732	166.31	31.54	0.889	45.59	Lossless
	3&4&5	26.84	0.767	134.47	32.50	0.899	36.50	Lossless
	4&5&6	27.33	0.795	120.16	32.93	0.904	33.82	Lossless

Table 10 Quality of the recovery for methods 1 and 2 (two levels)

Image quality of secret image 'Lena'					Image quality of secret image 'Pepper'				
Progressive thresholds (r_{j1}&r_{j2})	Recovery on level 1			Recovery on level 2	Progressive thresholds (r_{j1}&r_{j2})	Recovery on level 1			Recovery on level 2
	PSNR	SSIM	MSE			PSNR	SSIM	MSE	
2&3	28.53	0.843	91.12	Lossless	2&3	28.67	0.861	88.21	Lossless
3&4	29.08	0.859	80.26	Lossless	3&4	29.32	0.877	76.00	Lossless
4&5	29.63	0.871	70.80	Lossless	4&5	29.97	0.891	65.43	Lossless
5&6	29.85	0.874	67.30	Lossless	5&6	30.35	0.892	59.85	Lossless

$r_{j1}/(r_{j1} + r_{j2} + \ldots + r_{jk}) = 3/(3 + 4 + 5) = 25\%$ of the rearranged DCT data. The recovery on level 2 can recover about $(r_{j1} + r_{j2})/(r_{j1} + r_{j2} + \ldots + r_{jk}) = (3 + 4)/(3 + 4 + 5) = 7/12$ of the rearranged DCT data. Now, since

$$\begin{aligned} &2/(2+3+4) < 3/(3+4+5) < 4/(4+5+6) \\ &< 2/(2+3) < 3/(3+4) < 4/(4+5) < 5/(5+6) \\ &< [(2+3)/(2+3+4)] < [(3+4)/(3+4+5)] < [(4+5)/(4+5+6)], \end{aligned}$$

it is of no surprise that $MSE_{(4\&5\&6)} < MSE_{(3\&4\&5)} < MSE_{(2\&3\&4)}$ in Table 9, no matter in level 1's reconstruction or in level 2's. The inequality also implies that $MSE_{(5\&6)} < MSE_{(4\&5)} < MSE_{(3\&4)} < MSE_{(2\&3)}$ in Table 10. The same analysis also tells us that the second level reconstruction in Table 9 should be better than the first level reconstruction in Table 10, for $2/(2 + 3) < 3/(3 + 4) < 4/(4 + 5) < 5/(5 + 6) < [(2 + 3)/(2 + 3 + 4)] < [(3 + 4)/(3 + 4 + 5)] < [(4 + 5)/(4 + 5 + 6)]$. Likewise, the first level reconstruction in Table 10 should be better than the first level reconstruction in Table 9, for $2/(2 + 3 + 4) < 3/(3 + 4 + 5) < 4/(4 + 5 + 6) < 2/(2 + 3) < 3/(3 + 4) < 4/(4 + 5)$. The phenomenon is depicted in Figure 5.

The above analysis is for methods 1 and 2 (these two methods have identical shadows, and their recovered versions of secrets are also identical). Below we analyze method 3. We got Table 5 when we repeated the experiment in Section 4.3, using $k = 4$ keys to replace $k = 3$ keys (so the number of guardian stegos also increases from 3 to 4), and using $\{q_1 = 2, q_2 = 2, q_3 = 3,$ and $q_4 = 4\}$ to replace $\{q_1 = 2, q_2 = 2, q_3 = 3\}$. Since we used $(r, n) = (5, 6)$ in the secret sharing of [Cameraman, Lena], these two images cannot be viewed unless at least five stego images have been collected. On the other hand, $k = 4$ implies that the number of guardians is 4, and the number of non-guardians is $n - k = 6 - 4 = 2$. Hence, when five stegos are received, then either there are three guardians (together with two non-guardians), or there are four guardians (together with one non-guardian). Since $\{q_1 = 2, q_2 = 2, q_3 = 3,$ and $q_4 = 4\}$, the recovery is either of high level (because $q_4 = 4$), or of moderate level (because $q_3 = 3$). There is no way to get five stegos in which two are guardians and three are not; because the whole system only has two non-guardians. Consequently, only moderate level and high level are possible. Finally, if all six stegos are received, then only the lossless version exists. From this example, we can see that, even though the set $\{q_1 = 2, q_2 = 2, q_3 = 3,$ and $q_4 = 4\}$ has three distinct values, using $(r, n) = (n - 1, n) = (5, 6)$ makes the number of progressive levels drop from three to two. On the other hand, for the image House, because $(r, n) = (4, 6)$, so when people receive four stegos, the image House

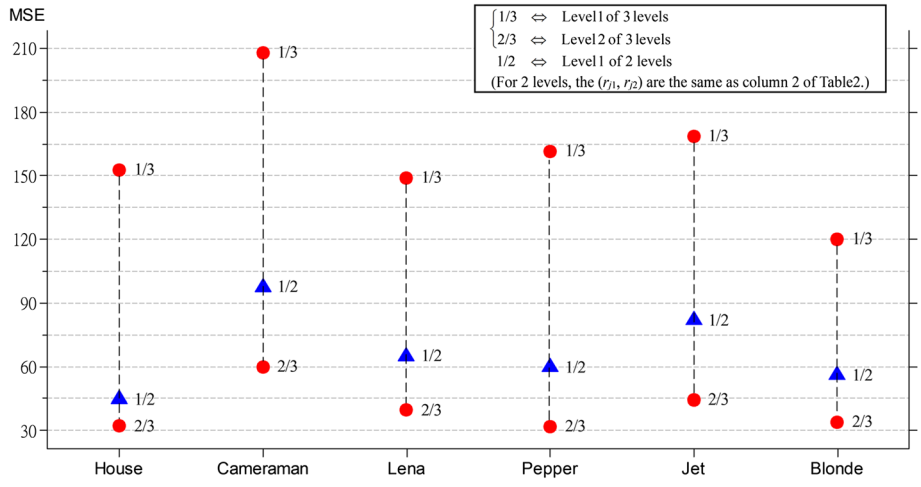

Figure 5 Secret recovery in methods 1 to 2. We copied data of Table 1 for three levels.

can be fully recovered (if four stegos are all guardians) or moderately recovered (if three of the stegos are guardians), or recovered with the lowest quality (if two of the four received stegos are guardians). Therefore, in method 3, although k is the number of type-q progressive parameters, the actual number of recovery levels might be lower than $k-1$, if the threshold value r of the image is very close to n.

Next we discuss the quality caused by the type-q parameters. Assuming that at least r stegos have been received, we can get the encrypted DCT of the image. Then, for $\{q_1 = 2, q_2 = 2, q_3 = 3, \text{and } q_4 = 4\}$, it means that regions $\{1, \ldots, i\}$ of the encrypted DCT can be decrypted if i of the received stegos are guardians ($i = 2, 3, 4$). Therefore, the recovery uses 2 (or 3, or 4) of the four regions of the DCT. For simplicity, assume that uniform partition was applied earlier to partition the DCT into regions. Then, on levels 1, 2 and 3, respectively, about 2/4, 3/4, 4/4 of the DCT information are utilized to recover the image. On the other hand, for the experiment done in Table 3 of Section 4.3, whose $k = 3$ parameters of type-q are $\{q_1 = 2, q_2 = 2, \text{and } q_3 = 3\}$, the information used on each level of image reconstruction is, respectively, about 2/3 and 3/3 of the DCT information. Since $2/4 < 2/3 < 3/4$, the recovery related to 2/4 should be worse than the recovery related to 2/3; whereas the recovery related to 2/3 should be worse than the recovery related to 3/4. In other words, the low-level recovery of Table 5 should be worse than the low-level recovery of Table 3, whereas the low-level recovery of Table 3 should be worse than the moderate-level recovery of Table 5. If we inspect Tables 3 and 5, we find that the result is really as expected.

6 Conclusions

In this paper, we proposed three kinds of progressive sharing methods to deal with multiple images. Method 1 is a basic progressive sharing method that decodes according to the sensitivity level of each secret image group. Method 2 is similar to method 1; however, in method 2, different weights are assigned to different cover image groups. Consequently, in the unveiling of the secret images, some groups of cover images are more useful than others. Method 3 uses a single threshold for each secret image with the same sensitivity level; however, the keys are progressively shared, making method 3 progressive. The increase in the shadow size caused by progressiveness can be neglected in method 3 because the size of the keys is much smaller than the size of the images.

We enhance conventional image sharing methods by providing the following features:

A. Multiple secret images are divided into several groups with distinct sensitivity levels, and each secret group has its own decoding thresholds for progressiveness. Note that each stego hides some information from 'all' groups of secret images. In other words, the information is integrated across the groups. In method 1, all stego JPEG codes have the same importance, as explained below: using 'any' r_{ji} of the n stegos, we can recover every secret image of secret group j; and the recovery quality is of the ith progressive level.
B. The cover images can also be divided into several groups, as introduced in method 2. Each cover group in method 2 has its own weight, and cover groups with more weight are more powerful (a cover group with weight = 3 is as powerful as three cover groups with weight = 1 each). Consequently, some covers have more influence than others in the revealing of secrets.
C. In method 3, in addition to using each secret group's own threshold to control its revelations, guardian stegos play an even more dominant role to control the revealing of secrets. Method 3 is also progressive.
D. We provide progressive decoding for multi-images, and the stegos are in JPEG format. Progressive decoding of multiple secret images is desirable in certain applications such as image retrieval from sensitive databases in crime investigation units, military departments, and the design team of a company.

By choosing a method that suits his/her purposes from our three proposed methods, a user can distribute his/her secret images among n stegos and obtain the progressive recovery effect that s/he desires.

Competing interests
The authors declare that they have no competing interests.

Acknowledgements
The work was supported by National Science Council of Republic of China, under grant NSC 100-2221-E-009-141-MY3. The authors also thank the reviewers for their valuable comments.

Received: 4 October 2014 Accepted: 7 January 2015
Published online: 14 February 2015

References

1. A Shamir, How to share a secret. Commun. ACM **22**(11), 612–613 (1979)
2. GR Blakley, Safeguarding cryptographic keys, in *Proceedings of AFIPS 1979 National Computer Conference, New Jersey, USA, vol. 48*, 1979, pp. 313–317
3. SK Chen, JC Lin, Fault-tolerant and progressive transmission of images. Pattern Recogn. **38**(12), 2466–2471 (2005)
4. RZ Wang, SJ Shyu, Scalable secret image sharing. Signal Process. Image Comm. **22**(4), 363–373 (2007)
5. KH Hung, YJ Chang, JC Lin, Progressive sharing of an image. Opt. Eng. **47**(4), 047006 (2008)
6. LST Chen, JC Lin, Multithreshold progressive image sharing with compact shadows. J. Electron. Imag. **19**(1), 013003 (2010)
7. WP Fang, Friendly progressive visual secret sharing. Pattern Recog. **41**(4), 1410–1414 (2008)

8. CC Thien, JC Lin, Secret image sharing. Comput. Graph. **26**(5), 765–770 (2002)
9. CC Lin, WH Tsai, Secret image sharing with steganography and authentication. J. Syst. Software **73**(3), 405–414 (2004)
10. CC Lin, WH Tsai, Secret image sharing with capability of share data reduction. Opt. Eng. **42**(8), 2340–2345 (2003)
11. FM Bui, D Hatzinakos, Biometric methods for secure communications in body sensor networks: resource-efficient key management and signal-level data scrambling. EURASIP J. Adv. Signal Process **2008**, 529879 (2008)
12. SS Lee, SF Hsu, JC Lin, Protection of PDF files: a sharing approach. Int. J. U-E- Serv Sci. Tech. **7**(2), 27–40 (2014)
13. D Wang, F Yi, On converting secret sharing scheme to visual secret sharing scheme. EURASIP J. Adv. Signal Process. **2010**, 782438 (2010)
14. TH Chen, KH Tsao, Threshold visual secret sharing by random grids. J. Syst. Software **84**(7), 1197–1208 (2011)
15. AT Boloorchi, MH Samadzadeh, T Chen, Symmetric threshold multipath (STM): an online symmetric key management scheme. Inform. Sci. **268**(1), 489–504 (2014)
16. KS Wu, A secret image sharing scheme for light images. EURASIP J. Adv. Signal Process. **2013**, 49 (2013)
17. LST Chen, SJ Lin, JC Lin, Reversible JPEG-based hiding method with high hiding-ratio. Int. J. Pattern Recogn. Artif. Intell. **24**(3), 433–456 (2010)
18. YY Tsai, DS Tsai, CL Liu, Reversible data hiding scheme based on neighboring pixel differences. Digit. Signal Process. **23**(3), 919–927 (2013)
19. S Mukherjee, AR Mahajan, Review on algorithms and techniques of reversible data hiding. Int. J. Res. Comput. Commun. Tech. **3**(3), 291–295 (2014)
20. J Lukas, J Fridrich, M Goljan, Digital camera identification from sensor pattern noise. IEEE Trans. Inform. Forensic Secur. **1**(2), 205–214 (2006)
21. E Brannock, M Weeks, R Harrison, *Watermarking With Wavelets: Simplicity Leads to Robustness* (Paper presented at the IEEE Southeastcon 2008, Huntsville, AL, 2008)
22. E Castillo, L Parrilla, A Garcia, U Meyer-Baese, G Botella, A Lloris, *Automated Signature Insertion in Combinational Logic Patterns for HDL IP Core Protection* (Paper presented at the 4th southern conference on programmable logic, San Carlos de Bariloche, 2008)
23. M Meenakumari, G Athisha, Improving the protection of FPGA based sequential IP core designs using hierarchical watermarking technique. J. Theor. Appl. Inf. Technol. **63**(3), 701–708 (2014)
24. I Orović, M Orlandić, S Stanković, An image watermarking based on the pdf modeling and quantization effects in the wavelet domain. Multimed. Tool. Appl. **70**(3), 1503–1519 (2014)
25. Z Wang, AC Bovik, HR Sheikh, EP Simoncelli, Image quality assessment: from error visibility to structural similarity. IEEE Trans. Image Process. **13**(4), 600–612 (2004)
26. Chi-square Steganography Test Program [http://www.guillermito2.net/stegano/tools/index.html]
27. N Provos, P Honeyman, Hide and seek: an introduction to steganography. IEEE Secur. Priv. **1**(3), 32–44 (2003)