International Journal of Pattern Recognition
and Artificial Intelligence
Vol. 23, No. 2 (2009) 263–285
© World Scientific Publishing Company

SECRET IMAGE SHARING: A BOOLEAN-OPERATIONS-BASED APPROACH COMBINING BENEFITS OF POLYNOMIAL-BASED AND FAST APPROACHES

KUN-YUAN CHAO* and JA-CHEN LIN

Department of Computer and Information Science
National Chiao Tung University
1001 Ta Hsueh Rd., Hsinchu, Taiwan, 300, R.O.C.
**kychao@cis.nctu.edu.tw*

In secret image sharing, a polynomial interpolation technique heavy experiences a computation load when the secret image is retrieved later. To the contrary, fast approaches often need larger storage space due to pixel expansion property. This paper proposes a missing-allowable (k, n) scheme which is fast and with a reasonable pixel expansion rate (per). The scheme generates n extremely-noisy shadow images for the given secret color image A, and any k out of these n shadows can recover A loss-freely. In average, to decode a color pixel of A, the retrieval uses only three exclusion-OR operations among 24-bit numbers. Hence, the new method has very fast decoding speed, and its pixel expansion rate is always acceptable $(0 < \text{per} < 2)$.

Keywords: Polynomial-style sharing; fast schemes; computation complexity; pixel expansion rate; exclusive-OR.

1. Introduction

Secret sharing using polynomials[1,9–11,13] is one of the popular approach to protect secret images. This kind of approach can restore the secret images without any loss, and the size of each shadow image can even be several times smaller than that of the given secret image.[9,11] Therefore, space-wasting is seldom a problem for sharing using polynomials. However, the retrieval computation is very slow because of the evaluation of polynomials.

On the other hand, a faster approach is to use the digitalized versions of Refs. 3, 4, 15 and 16 to share a digital image among several "size-enlarged" digital images called shadows. Recently, to improve the efficiency and speed in sharing digital color images, Lukac and Plataniotis smartly proposed some easily implemented methods[5–8] whose decoding use "OR-like" operations or look up basis matrices.

*Author for correspondence

Their new methods can recover the original image error-freely at a very fast speed; although in Refs. 5–7 quite often the shadow images generated in their (k, n)-schemes are still several times larger than the secret image. (Notably, (k, n)-schemes mean that in the reconstruction of the secret, any k out of n shadows can obtain the secret; while less than k shadows cannot.) As for Ref. 8, the size of each shadow image is the same as the secret image size, but Ref. 8 considers $k = n = 2$ only.

Although people can use the digitalized version of an elegant method proposed in Ref. 14 that has no size expansion because of the probabilistic skill; the recovered secret image is not lossless. In general, size expansion problem is a disadvantage for fast approaches: to store digital shadow images in the computer often requires large storage space.

From the analysis in the above two paragraphs, we can see that these two types of sharing approaches are quite different, and each has its own speed-versus-space advantage and disadvantage. A question arises naturally: "Can a sharing system have both advantages in speed and space?" In other words, can people have some economic-size shadows which can reconstruct the given secret image in a loss-free manner after only using a few operations to decode each pixel? The answer is positive. Wang *et al.*[12,17] had gracefully provided their answer, to a certain level, in their second scheme[12] which is an (n, n) scheme.

In the current paper, we will improve Wang *et al.*'s (n, n) scheme in order to have the "missing-allowable" (k, n)-threshold ability, i.e. in the reconstruction of the secret, any k out of n shadows will work. The proposed scheme generates the n desired shadows for a given color image A, so that each shadow's size is less than two times the size of A. Furthermore, the lossless decoding process only uses a few exclusive-OR (XOR) operations (symbolized as "$\oplus$"), so there is no complex computation.

The remaining portion of the paper is organized as follows. Section 2 briefly reviews some polynomial-style and fast schemes for image sharing. Section 3 presents the proposed method. Section 4 gives some analyses about the proposed scheme. Experimental results are included in Sec. 5. Finally, conclusions are provided in Sec. 6.

2. A Simple Review of Image-Sharing Methods

This section first review two kinds of well-known techniques for sharing secret images: polynomial-style approaches[1,9–11,13] as described in Sec. 2.1, and fast as approaches[5–8] as in Sec. 2.2. In addition, Sec. 2.3 briefly describes Wang *et al.*'s second scheme in Ref. 12 based on Boolean operations.

2.1. *Polynomial-style schemes*

All schemes in Refs. 1, 9–11 and 13 apply the polynomial interpolation to divide a secret data A into n distinct data sets $D_1, D_2, \ldots, D_n$ called shares or shadows; and the secret data A cannot be revealed until k of n shadows become available. To share an image, the data A becomes the values of pixels. To split A into n shadows, people can pick a prime number p and a polynomial

$$q(x) = (a_0 + a_1 x + \cdots + a_{k-1}x^{k-1}) \bmod p$$

of degree $k-1$ in which a_0 is the data A, and all other coefficients $a_1, a_2, \ldots, a_{k-1}$ are randomly chosen from an integer in 0 to $(p-1)$. Then evaluate

$$D_1 = q(1), \ldots, D_i = q(i), \ldots, D_n = q(n).$$

Using any k pairs of n produced pairs $\{(i, D_i)\}_{i=1}^{n}$, people can get all coefficients $a_1, a_2, \ldots, a_{k-1}$ in $q(x)$ by the Largrange's interpolation, and hence the secret data $A = a_0$ is also revealed. To reveal the secret data A, the computation complexity is $O(k \log^2 k)$ for polynomial interpolation.

2.2. *Lukac and Plataniotis's fast schemes*

For fast decoding, digitalized versions based on Refs. 3, 4, 15 and 16 can be used. However, to share digital color images more effectively, Lukac and Plataniotis elegantly restructure the original digital color image files using the "OR-like" function or looking up basis matrices in their sharing methods.[5–8]

Their new schemes to share and recover digital images are easy to implement, and the retrieval speeds are very fast; although in Refs. 5–7, the shadow images generated in their (k, n)-schemes are still several times larger than the secret image. The problem might get worse as the values of k and n become very large. (As for Ref. 8, as stated earlier, the size of each shadow image is the same as the secret image size, but Ref. 8 considers $k = n = 2$ only.) As a result, to store the created digital shadows often need larger storage space in a computer.

2.3. *Wang et al.'s fast* (n, n) *scheme*

Wang *et al.* also proposed in Refs. 12 and 17 some fast schemes with the intention of small pixel expansion rate (per). Their (k, n) scheme in Ref. 17 and their first scheme (a $(2, n)$ scheme) in Ref. 12 are both *probabilistic* (and hence might cause loss in image retrieval). Their second scheme in Ref. 12 is a *deterministic* (n, n) scheme for grayscale images (extension to color images is also possible); and hence causes lossless retrieval. Notably, in their (n, n) scheme,[12] it splits a secret image A among n shadows $C_1, C_2, \ldots, C_n$, whose pixel expansion rate is one. After receiving all n shadows, it uses only $n-1$ XOR operations to reconstruct a pixel of A. Their (n, n)-scheme algorithm is as follows:

Coding:

Step 1. Input a secret image A.
Step 2. Generate n-1 random images $B_1, B_2, \ldots, B_{n-1}$, each has a size A.

Step 3. Compute the shadows as follows:

$$\begin{aligned} C_1 &= B_1, \\ C_2 &= B_1 \oplus B_2, \\ &\vdots \\ C_{n-1} &= B_{n-2} \oplus B_{n-1}, \\ C_n &= B_{n-1} \oplus A. \end{aligned}$$

Step 4. Output the n shadows $C_1, C_2, \ldots, C_n$.

Decoding:

Reveal A using the formula $A = C_1 \oplus C_2 \oplus \cdots \oplus C_n$.

In this paper, in order to extend Wang *et al.*'s (n, n) no-threshold scheme to $(k,\ n)$ threshold scheme; we introduce a $(k,\ n, m)$ shadows-assignment matrix H, and a $\{B_1, B_2\}$ partition-and-recombination process. The scheme still holds the two advantages of Ref. 12: fast computation speed and small pixel expansion rate. In fact, we only need three XOR operations in average to reconstruct a pixel; and the ratio of each shadow's size over the secret image's size is between 0 and 2, i.e. $0 <$ per < 2 (and per $= 2/n$ in $k = n$ case). The statement is true in all $(k,\ n)$ cases.

3. The Proposed Method

To generate the desired shadows, the encoding algorithm in Sec. 3.1 will need the two new techniques described in Secs. 3.2 and 3.3. To help readers understand the encoding, a numerical example is also given in Sec. 3.4.

Then, Sec. 3.5 introduces the decoding algorithm that retrieves the secret. For easier understanding of the decoding algorithm; a numerical example for decoding is also given in Sec. 3.6.

3.1. *The encoding algorithm*

First, we illustrate here our encoding algorithm which creates n final shadows that meet the $(k,\ n)$ threshold goal. This encoding algorithm will use two other new tools: the $(k,\ n,\ m)$ shadows-assignment matrix in Sec. 3.2, and the {B1, B2} partition-and-recombination process in Sec. 3.3.

The encoding algorithm:

Step 1. Input a color secret image A.
Step 2. Generate a random image B_1 so that B_1 and A have the same size.
Step 3. Generate another image B_2 using $B_2 = B_1 \oplus A$, where $\oplus$ denotes bit-by-bit XOR.
Step 4. Let $m = C^n_{k-1}$. Generate the $(k,\ n,\ m)$ shadows-assignment matrix H described in Sec. 3.2. Notably, the matrix H is public.

Step 5. Use the two images B_1 and B_2 to generate m temporary shadows $C_1, C_2, \dots, C_m$ by using the {B1, B2} partition-and-recombination process (see Sec. 3.3 and Fig. 1).

Step 6. Assign the duplicated copies of the m temporary shadows $C_1, C_2, \dots, C_m$ to the n persons according to the shadows-assignment matrix H mentioned in Step 4 (see Sec. 3.2 for more details of the assignment). For each person i, the final shadow D_i that he has is exactly the union of those copies assigned to him.

To understand the above encoding algorithm, see the example in Sec. 3.4.

Remark. Each participant gets several temporary shadows which are all random matrixes. Thus, it is better to have a discussion about how to distinguish the temporary shadows so that the related temporary shadows inside each final shadow can be distinguished easily later to recover the image.

Option 1. Assume that, according to the shadows-assignment matrix H, there is a person who owns three temporary shadows $\{C_2, C_3, C_6\}$. Let us use this person as an example. If the size of each temporary shadow is $w \times h$, then, before inserting the separator, the size of this person's final shadow was $3w \times h$. (The first w rows were C_2, next w rows were C_3, final w rows were C_6.) So the number of rows of the final shadow was three times larger than that of each temporary shadow, but the columns (h) were the same. Now, after the first w rows, since we have already completed C_2, and C_3 is to be attached behind C_2, we insert a separator-row of $(h/2) + (h/2) = h$ elements, i.e. 2222222222222222222233333333333333333333, so that people can understand C_2 is above this separator-row and C_3 is below

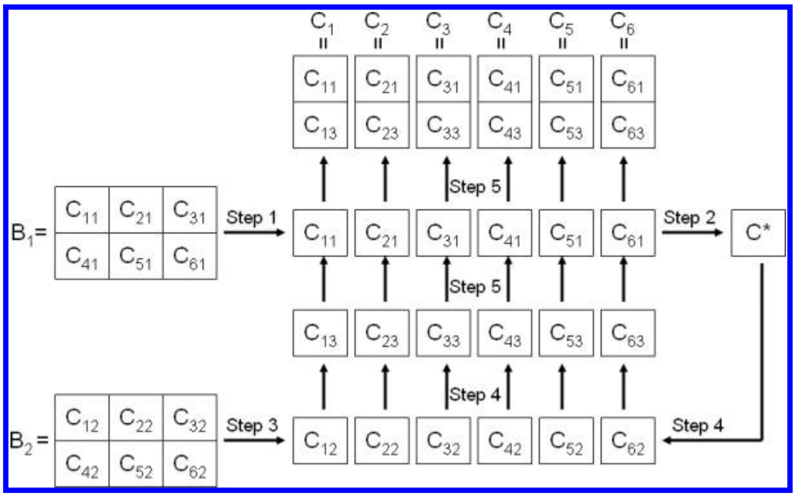


Fig. 1. A flowchart showing the process that transforms $\{B_1, B_2\}$ to $\{C_1, C_2, \dots, C_6\}$. In this example, $(k, n) = (3, 4)$; so $m = C^n_{k-1} = 6$ accordingly.

this separator-row. Then we store the C_3 using next w rows. Then, insert another separator-row of $h/2 + h/2 = h$ elements, i.e.

3333333333333333333336666666666666666666666, before attaching C_6. In summary, if separators are used, the final-shadow has $(3w + 2)$ rows rather than $3w$ rows, and the $(3w + 2)$ rows owned by this $\{C_2, C_3, C_6\}$ person will be

$[C_2]$ (which has w rows, each row has h pixels)
222222222222222222222333333333333333333333
$[C_3]$ (which has w rows, each row has h pixels)
3333333333333333333336666666666666666666666
$[C_6]$ (which has w rows, each row has h pixels)

Notably, in Step 4 of the encoding algorithm, we have already stated that the shadows-assignment matrix H is public, so the decoder can always read from H to know the number of temporary shadows owned per participant. For example, in H in Eq. (3), each participant gets $1 + 1 + 1 = 3$ temporary shadows. So, in Option 1, the decoder will know that the number of rows in this final shadow is $3w + (3 - 1) = (3w + 2)$. Therefore, the decoder can always figure out how many rows are in the final shadow, and hence, know how many pixels are in each row.

Option 2. (An option using the convention of ascending-order indices). In fact, from the viewpoint stated in the final paragraph of Option 1 above, the separator rows can also be omitted, as explained below. Assume each participant owns certain shadows. Let the shadow indices be all arranged in the ascending order. For example, if the matrix H is as shown in Eq. (3), then the person P_1 owns (copies of the) temporary shadows C_4, C_5, C_6, the person P_2 owns temporary shadows C_2, C_3, C_6, the person P_3 owns C_1, C_3, C_5, and the person P_4 owns C_1, C_2, C_4. Notice the indices are all in ascending order (i.e. $4 < 5 < 6$; $2 < 3 < 6$; $1 < 3 < 5$; $1 < 2 < 4$). So, even if we do not use separators, the decoder can still read the "public" matrix H to know that each participant owns three temporary shadows; and hence, divide each person's final shadow into three parts of equal size; and then use matrix H to identify easily which temporary shadow is the first one-third of that person's final shadow, which temporary shadow is the middle one-third, and which temporary shadow is the final one-third.

3.2. *The (k, n, m) shadows-assignment matrix H (which has n rows and $m = C^n_{k-1}$ columns)*

To design a threshold (k, n) scheme, we may first directly utilize Wang *et al.*'s nonthreshold (m, m) scheme for some carefully chosen parameter

$$m = C^n_{k-1}.$$

(The reason why m is chosen as $m = C^n_{k-1}$ will be explained later.) So, Wang *et al.*'s (m, m) method gives us $m = C^n_{k-1}$ shadows (these are not our final shadows, just consider them as our temporary shadows). Then, we duplicate each temporary shadow several times. Then, for n people participating in the sharing game, let each person get one or no copy from each of m temporary shadows. Each person can have copies from more than one temporary shadow. However, no person can get copies from all m shadows; otherwise, that person alone can unveil the secret.

After this distribution assignment of the copies of m produced temporary shadows, we wish that when any k or more people gather together in an image-recovery meeting, the chairman of the meeting can collect all m temporary shadows from the attendants of this meeting; and hence, can restore the secret image according to Wang *et al.*'s (m, m) image-recovery scheme. We also require that a meeting of less than k people together is insufficient to collect all m temporary shadows; and hence, cannot reveal the secret image. We will call the two requirements stated above in this paragraph as the "(k, n, m) shadows-assignment requirements".

From the idea above, we may create a matrix H of n rows and m columns. Its n rows represent n persons; and its m columns represent the m (distinct) temporary-shadows produced by Wang *et al.*'s deterministic (m, m) scheme. The element of H is either 0 or 1. The ith person (row) has a copy of jth shadow image (column) if and only if $H_{ij} = 1$. In order to make the matrix meet the expected (k, n, m) shadows-assignment requirements described above, we let each column of H have exactly $k-1$ *zero*s and $n-k+1$ *one*s. More specifically, let H have $m = C^n_{k-1}$ columns, and each column of H be a permutation of the n-dimensional basic column vector (000...0011111...111) which has $k-1$ leading zeros followed by $n-k+1$ ones.

This obviously guarantees that: (i) each temporary shadow C_j will appear at least once when k out of n persons attend the image-recovery meeting; (ii) at least one temporary shadow C_j will disappear when $k-1$ or fewer persons attend the recovery meeting. The proof is as follows:

Proof. Consider the equation

$$\begin{bmatrix} P_1 \\ P_2 \\ \vdots \\ P_n \end{bmatrix}^T \times \begin{bmatrix} H_{11} & H_{12} & \cdots & H_{1m} \\ H_{21} & H_{22} & \cdots & H_{2m} \\ \vdots & \vdots & \vdots & \vdots \\ H_{n1} & H_{n2} & \cdots & H_{nm} \end{bmatrix} = \begin{bmatrix} X_1 \\ X_2 \\ \vdots \\ X_m \end{bmatrix}^T, \tag{1}$$

where P_i and $H_{ij} \in \{0,1\}$ $(i = 1,2,\ldots,n; j = 1,2,\ldots,m)$. In this equation, P_i represents the attendance status of ith person (0 is absence and 1 is attendance); and H is the created (k, n, m) shadows-assignment matrix. Therefore,

$$X_j = P_1 \times H_{1j} + P_2 \times H_{2j} + \cdots + P_n \times H_{nj} \tag{2}$$

counts the number of times (copies) that the temporary shadow C_j appears in the image-recovery meeting. Two observations are:

(i) When k persons attend the recovery meeting, then k of n elements in $(P_1, \ldots, P_n)$ are one, and the remaining $n - k$ elements are zero. Therefore, each X_j must be at least one, for there is exactly $k - 1$ *zero*s in every column j of H. This implies that each temporary shadow C_j will appear at least once in the recovery meeting.

(ii) When only $k-1$ or fewer persons attend the recovery meeting, then at most $k-1$ of the n elements in $(P_1, \ldots, P_n)$ are one; or equivalently, at least $n-k+1$ of n elements in $(P_1, \ldots, P_n)$ are zero. Let $\mathrm{Col}_j = (H_{1j}, H_{2j}, \ldots, H_{nj})$ be a column of H whose $n-k+1$ *one*s happen to appear at the positions where the vector $(P_1, \ldots, P_n)$ obtained these (at least) $n-k+1$ *zero*s. (If $(P_1, \ldots, P_n)$ has more than $n-k+1$ *zero*s, then just randomly choose $n-k+1$ positions from the zero entries of $(P_1, \ldots, P_n)$.) The inner product of the vector $(P_1, \ldots, P_n)$ and this special Col_j will be zero. In other words, $X_j = 0$. So the temporary shadow C_j disappears in the recovery meeting. □

In the above construction of the matrix H, recall that we let all $m = C^1_{k-1}$ permutations of the n-dim vector $(000\ldots001111\ldots11)$, which has exactly $k-1$ leading *zero*s and $n-k+1$ *one*s, be used as the m columns; and thus obtain the expected n-by-m matrix H. Hereinafter, the matrix H will be called the "(k, n, m) shadows-assignment matrix".

Below is an example showing the (k, n, m) shadows-assignment matrix H. Assume $(k, n) = (3, 4)$, so $m = 6 = C^4_{3-1}$. Note that each column is just a permutation of the first column, and the first column is an $n = 4$-dimensional vector which has exactly $k - 1 = 3 - 1 = 2$ zeros.

$$H = \begin{array}{c} \\ P_1 \\ P_2 \\ P_3 \\ P_4 \end{array} \begin{array}{c} \begin{array}{cccccc} C_1 & C_2 & C_3 & C_4 & C_5 & C_6 \end{array} \\ \begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 \end{bmatrix} \end{array}. \tag{3}$$

As a result, H has four rows (since $n = 4$) and six columns (since $m = 6 = C^4_{3-1}$). In this example, the person P_1 owns (copies of the) temporary shadows C_4, C_5, C_6, the person P_2 owns temporary shadows C_2, C_3, C_6, the person P_3 owns temporary shadows C_1, C_3, C_5, and the person P_4 owns temporary shadows C_1, C_2, C_4. In this shadows-assignment process, any $k = 3$ people gathered together can guarantee the appearance of all six temporary shadows C_1, C_2, C_3, C_4, C_5 and C_6; but less than three persons cannot. In other words, three or more people can recover the secret image according to Wang *et al.*'s deterministic (6, 6) scheme using these six temporary shadows. Less than three people cannot recover because some C_j disappears.

3.3. *Partition-and-recombination process of* $\{B_1, B_2\}$

In Sec. 3.2, after assigning m temporary shadows to n people according to the matrix H, each person gets some temporary shadows. Each person i can combine the temporary shadows that he holds into a single shadow D_i specially designed for him. Then these n final shadows $D_1, D_2, \ldots, D_n$ owned respectively by these n persons are the final output of a very simple (k, n)-threshold scheme.

This simplest design is easy (it only needs the idea of using H as mentioned in Sec. 3.2 above, and matrix H itself is easy to construct). However, according to Wang *et al.*'s deterministic (m, m) scheme, all m temporary shadows have the same size as that of secret image A. This often causes space-and-speed inefficiency problem. More specifically, as becomes larger, this very simple (k, n)-threshold scheme will have two drawbacks: (1) big-size problem for each D_i of the final shadows $\{D_1, D_2, \ldots, D_n\}$; and (2) many XOR operations in decoding. In order to avoid these two drawbacks, we do not use Wang *et al.*'s output as the m temporary shadows. Instead, we create our own m temporary shadows. This can be done by the two-shadows partition-and-recombination preprocess proposed below.

First, create a random image B_1 whose size is identical to that of the secret image A. Then, generate another same-size image $B_2 = B_1 \oplus A$ using XOR in a bit-by-bit manner. Notably, according to the inverse property of XOR operation, secret image A can be recovered by the equation $A = B_1 \oplus B_2$. Then, create m temporary shadows $C_1, C_2, \ldots, C_m$ by partitioning and recombining B_1 and B_2, as follows [see Fig. 1 for an example using $(k = 3,\ n = 4)$]:

Step 1. Randomly generate an image B_1 whose size is identical to A's. Then partition B_1 into $m = C^n_{k-1}$ nonoverlapping blocks $C_{11}, C_{21}, \ldots, C_{m1}$. The upper half of each temporary shadow C_i $(1 \leqq i \leqq m)$ is the block C_{i1} contained in B_1.

Step 2. Create the security mask C^*, which is also a block, by the XOR equation

$$C^* = C_{11} \oplus C_{21} \oplus \cdots \oplus C_{m1}.$$

Step 3. Create an image $B_2 = B_1 \oplus A$ using XOR in a bit-by-bit manner. (A, B_1 and B_2 have the same size.) Then partition B_2 into m nonoverlapping blocks $C_{12}, C_{22}, \ldots, C_{m2}$.

Step 4. For security reason, shift each C_{i2} to C_{i3} by the formula $C_{i3} = C_{i2} \oplus C^*$.

Step 5. After physically attaching each C_{i3} to C_{i1}, we obtain m temporary shadows $C_1, C_2, \ldots, C_m$. (Notably, the upper half of each C_i is C_{i1}, and the lower half of each C_i is C_{i3}.)

As a remark, if B_1 and B_2 cannot be divided equally into m blocks of the same size, some redundant pixels can be filled in B_1 and B_2. In the $(k, n) = (3, 4)$ example, if the size of A is 512×512, then B_1 and B_2 need two redundant pixels respectively, because 512×512 is not a full multiple of $m = C^n_{k-1} = 6$.

In the inverse process to obtain B_1 and B_2 from $C_1, C_2, \ldots, C_m$, the algorithm is as follows (see Fig. 2 where we still use ($k = 3$, $n = 4$) as an example):

Step 1. Extract m nonoverlapping blocks $C_{11}, C_{21}, \ldots, C_{m1}$ which are the upper half of $C_1, C_2, \ldots, C_m$, respectively.
Step 2. Recover the security mask C^* by the equation $C^* = C_{11} \oplus C_{21} \oplus \cdots \oplus C_{m1}$.
Step 3. Recover the random image B_1 by physically attaching $C_{11}, C_{21}, \ldots, C_{m1}$ to each other.
Step 4. Extract the m nonoverlapping blocks $C_{13}, C_{23}, \ldots, C_{m3}$ which are the lower half of $C_1, C_2, \ldots, C_m$, respectively.
Step 5. Recover the m blocks $C_{12}, C_{22}, \ldots, C_{m2}$ using the shift-back equation $C_{i2} = C_{i3} \oplus C^*$ (where $1 \leqq i \leqq m$).
Step 6. Recover the image B_2 by physically attaching $C_{12}, C_{22}, \ldots, C_{m2}$ to each other.
Step 7. Recover the secret image A by the equation $A = B_1 \oplus B_2$.

3.4. *Numerical example of encoding example*

In the following encoding example, we do it step by step. Without the loss of generality, assume ($k = 3$, $n = 4$), so $m = C^n_{k-1} = 6$. Also, for easier description, we just use gray-values rather than color-values in the example.

Step 1. Assume the given secret image is a 2×3 image $A = \begin{bmatrix} 55 & 76 & 186 \\ 67 & 133 & 202 \end{bmatrix}$, which has six pixel values.
Step 2. Randomly generate an image B_1 whose size is identical to A's. For example, randomly let $B_1 = \begin{bmatrix} 149 & 225 & 41 \\ 93 & 210 & 32 \end{bmatrix}$.

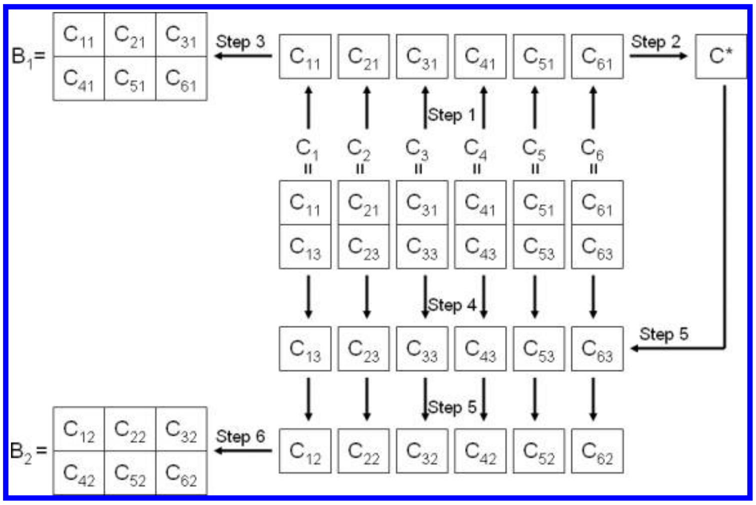


Fig. 2. A flowchart of the inverse process to recover B_1 and B_2 from $C_1, C_2, \ldots, C_m$. In this example, $(k, n) = (3, 4)$; so $m = C^n_{k-1} = 6$ accordingly.

Step 3. Generate another image B_2 by applying bit-by-bit XOR to B_1 and A, i.e.

$$B_2 = B_1 \oplus A = \begin{bmatrix} 162 & 173 & 147 \\ 30 & 87 & 234 \end{bmatrix}$$

where $162 = 55 \oplus 149, 173 = 76 \oplus 225$, etc.

Step 4. According to the skill in Sec. 3.2, generate a ($k = 3$, $n = 4$) threshold shadows-assignment matrix

$$H = \begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 \end{bmatrix}$$

which has $n = 4$ rows and $m = C^n_{k-1} = 6$ columns. Note that each column is a permutation of the first column vector 0011.

Step 5. Firstly, use bit-by-bit XOR on the elements of B_1 to obtain the security block $C^* = [242]$ by the formula $[149] \oplus [225] \oplus [41] \oplus [93] \oplus [210] \oplus [32] = [242]$.

Then, according to Sec. 3.3, generate six temporary shadows

$$C_1 = \begin{bmatrix} 149 \\ 80 \end{bmatrix}, \quad C_2 = \begin{bmatrix} 225 \\ 95 \end{bmatrix}, \quad C_3 = \begin{bmatrix} 41 \\ 97 \end{bmatrix}, \quad C_4 = \begin{bmatrix} 93 \\ 236 \end{bmatrix},$$
$$C_5 = \begin{bmatrix} 210 \\ 165 \end{bmatrix}, \quad C_6 = \begin{bmatrix} 32 \\ 24 \end{bmatrix}$$

where the six lower halves are the result of transforming the six lower halves of B_2, by doing bit-by-bit XOR with $C^* = [242]$. For example, $80 = 162 \oplus 242$, and $95 = 173 \oplus 242$.

Step 6. According to the assignment matrix H, assign the copies of the $m = 6$ temporary shadows $\{C_1, C_2, \ldots, C_6\}$ to the $n = 4$ persons. So, our $n = 4$ final shadows, hold by $n = 4$ persons respectively, are

$$D_1 = \begin{bmatrix} 93 & 210 & 32 \\ 236 & 165 & 24 \end{bmatrix}, \quad D_2 = \begin{bmatrix} 225 & 41 & 32 \\ 95 & 97 & 24 \end{bmatrix},$$
$$D_3 = \begin{bmatrix} 149 & 41 & 210 \\ 80 & 97 & 165 \end{bmatrix}, \quad D_4 = \begin{bmatrix} 149 & 225 & 93 \\ 80 & 95 & 236 \end{bmatrix}$$

where D_1 and D_2 both have a copy of the temporary shadow $C_6 = \left[\begin{smallmatrix} 32 \\ 24 \end{smallmatrix}\right]$.

3.5. *The decoding algorithm*

Given any k final shadows, for example $\{D_1, D_2, \ldots, D_k\}$, out of n final shadows produced in Step 6 of Sec. 3.1, the secret image A can be restored as follows:

The decoding algorithm:

Step 1. After referring to (k, n) shadows-assignment matrix H generated in Step 4 of the encoding algorithm, we know which temporary shadows in

$\{C_1, C_2, \ldots, C_m\}$ are included in each final shadow D_i. Therefore, all m temporary shadows $C_1, C_2, \ldots, C_m$ can be extracted from these k final shadows. (For the reason, the reader can refer to (k, n, m) shadows-assignment requirements in Sec. 3.2, and the proof near Eq. (2).

Step 2. Use all m temporary shadows $C_1, C_2, \ldots, C_m$ to generate B_1 and B_2 by implementing the inverse process of $\{B_1, B_2\}$-partition-and-recombination process (see Sec. 3.3 and Fig. 2).

Step 3. Reveal the secret image A using $A = B_1 \oplus B_2$.

Remark. Step 1 above stated that we are able to know which temporary shadows in $\{C_1, C_2, \ldots, C_m\}$ are included in each final shadow D_i. As for how to distinguish the temporary shadows in each final shadow D_i (so that the related temporary shadows inside each final shadow D_i can be distinguished easily to recover the image), see the Remark at the end of Sec. 3.1.

3.6. *Numerical example for decoding*

In the following decoding example, still assume ($k = 3, n = 4$). So, decoding requires any three of the four final shadows. Without the loss of generality, assume D_1, D_2, D_3 are the three available shadows.

Step 1. With the help of matrix H in Step 4 of encoding process, we extract all $m = C^n_{k-1} = 6$ temporary shadows $C_1, C_2, \ldots, C_6$, which are the same as those created in Step 5 of encoding process of Sec. 3.4.

Step 2. Recover B_1 and B_2, which are the same as those in Steps 2 and 3 of the encoding process, by implementing the inverse process of $\{B_1, B_2\}$ partition-and-recombination process to all six temporary shadows $C_1, C_2, \ldots, C_6$ (see Fig. 2).

Step 3. Reveal the secret image A by $A = B_1 \oplus B_2 = \begin{bmatrix} 55 & 76 & 186 \\ 67 & 133 & 202 \end{bmatrix}$.

4. Analysis

4.1. *Recoverability and security*

In general, each (k, n) threshold secret sharing scheme must satisfy both requirements: the recoverability (any k or more shadows can reveal all information of A) and the security (any $k - 1$ or fewer shadows cannot reveal secret image A).

In our scheme, when any k out of n final shadows are gathered (for example, $D_1, D_2, \ldots, D_k$), secret image A is revealed by Steps 1–3 of the decoding algorithm. These steps also explain why our scheme satisfies the recoverability requirement. Firstly, if k or more final shadows are gathered, then we can extract all m temporary shadows $C_1, C_2, \ldots, C_m$ from k available final shadows according to (k, n, m) shadows-assignment requirements of matrix H. Secondly, after physically dividing each C_i into upper half C_{i1} and lower half C_{i3}, we can get C^* which is defined by $C^* = C_{11} \oplus C_{21} \oplus \cdots \oplus C_{m1}$. Then, we can restore $C_{12}, C_{22}, \ldots, C_{m2}$ using $C_{i2} = C_{i3} \oplus C^*$ for each $i = 1, \ldots, m$. Then recover $B_1 = \{C_{11}, C_{21}, \ldots, C_{m1}\}$ and $B_2 = \{C_{12}, C_{22}, \ldots, C_{m2}\}$. Therefore, the secret image A can be revealed using $A = B_1 \oplus B_2$.

Our scheme also satisfies the security requirement. Assume that only $k - 1$ or fewer final shadows are available. Then, according to $(k,\ n,\ m)$ shadows-assignment requirements of matrix H, people cannot obtain all m temporary shadows $C_1, C_2, \ldots, C_m$ from these final shadows (see the proof (ii) below Eq. (2) in Sec. 3.2). Assume C_q is missing. As a result, people cannot obtain C^* defined by $C^* = C_{11} \oplus C_{21} \oplus \cdots \oplus C_{m1}$, due to the lack of C_{q1} which is the upper half of C_q. Then, without C^*, people cannot restore $C_{12}, C_{22}, \ldots, C_{m2}$ defined by $C_{i2} = C_{i3} \oplus C^*$ $(1 \leqq i \leqq m)$. So, people cannot generate B_2. As a result, secret image $A = B_1 \oplus B_2$ cannot be revealed due to the absence of B_2.

Below we discuss the probability of obtaining the right secret image A through guessing. Without the loss of generality, assume that a betrayal party of $k - 1$ persons has already gathered $m - 1$ temporary shadows $C_1, C_2, \ldots, C_{m-1}$ without C_m. Notably, $A = \{A_i | 1 \leqq i \leqq m\}$, i.e. image A can be divided to m blocks, and the recovery of A can be done block by block; in other words, since $A = B_1 \oplus B_2$, we have

$$A_i = C_{i1} \oplus C_{i2} = C_{i1} \oplus (C_{i3} \oplus C^*) = C_{i1} \oplus C_{i3} \oplus (C_{11} \oplus C_{21} \oplus \cdots \oplus C_{m1}), \quad 1 \leqq i \leqq m.$$

Because of the lack of $C_m = [C_{m1} | C_{m3}]^T$, the betrayal party will have to guess a value for a pixel in C_{m1}, then they can use this guessing value to obtain a set of $m - 1$ pixels' values (one value per block in $A_1, A_2, \ldots, A_{m-1}$). Then they need to guess the value of a pixel at the corresponding position of C_{m3} (or A_m) so that the pixel value at that position of A_m can also be shown. The above is just to recover a pixel (for example, the top-leftmost pixel) of each block A_i, $1 \leqq i \leqq m$. This value-guessing of two pixels will be repeated bksize times. Here, bksize is the size of each block A_i $(1 \leqq i \leqq m)$; hence bksize is m times smaller than the image size of A.

From the description above, we can evaluate the probability of obtaining the right color image A with size $w \times h$ as follows. (For illustration, still assume $(k, n) = (3, 4)$; so $m = C^n_{k-1} = 6$ accordingly.)

$$\text{Probability} = s^{\text{bksize1}} \times s^{\text{bksize3}} = \left(\frac{1}{\text{pixelscale}}\right)^{\frac{w \times h}{m}} \left(\frac{1}{\text{pixelscale}}\right)^{\frac{w \times h}{m}} = \left(\frac{1}{2^{24}}\right)^{\frac{w \times h}{m} \times 2}$$

which is

$$\left(\frac{1}{2^{24}}\right)^{512 \times 512/3} = \left(\frac{1}{2^{24}}\right)^{87381} = 10^{-631304}$$

if the image size is $w \times h = 512 \times 512$. Here, $s = 1/2^{24}$ is the probability to guess successfully a pixel's value; bksize1 is the number of pixels in C_{m1}; and bksize3 is the number of pixels in C_{m3}. To improve the security further, people can use a prime number as a key (a seed) of a random number generator to rearrange the pixel positions in secret image A (as we did in Ref. 9) before encoding.

4.2. *Time complexity and storage space needed*

In terms of computation complexity, Thien and Lin's polynomial sharing scheme[9] needs $\mathrm{O}(\log^2 k)$ mathematical operations to reveal a pixel. Although Wang and Su[11] reduced 40% in size of Thien and Lin's shadow images, their scheme still need $\mathrm{O}(\log^2 k)$ mathematical operations to reveal a pixel. As for the digitalized versions derived from Refs. 3, 4, 15 and 16, they need $O(k \times \text{per})$ OR operations to reveal a pixel. Here, the value k means that the secret-recovery requires k gathered shadows, and the value of per represents pixel expansion rate (per $\geqq 2$ in Refs. 3, 4, 15 and 16). Lukac and Plataniotis's elegant methods[5–7] also requires $O(k \times \text{per})$ "OR-like" operations to restore an original input pixel in (k, n)-threshold schemes. Lukac and Plataniotis's special method[8] requires only 1 B-bit "OR-like" operation to restore a pixel of B-bit color secret image, but Ref. 8 only deals with the $k = 2 = n$ scheme.

Fang and Lin[2] proposed two other SS (sharing schemes), i.e. an (n, n) XOR-SS and a (k, n) OR-SS, to reduce the size of shadows in Lukac and Plataniotis's.[7] But the (k, n) OR-SS scheme in Ref. 2 still needs many OR operations in decoding, and the complexity is similar as that of Wang *et al.*'s (k, n) colored probabilistic scheme.[17] In Wang *et al.*'s (n, n) scheme,[12] only $n - 1$ XOR operations are required to reconstruct each pixel, which is the same as Fang and Lin's (n, n) XOR-SS scheme.[2] Obviously, the decoding time of most inventions above increases as the value of k or n increases.

For this concern, our new scheme tries to make more stable the speed of Wang *et al.*'s[12] for any n. In any (k, n) threshold cases, no matter how large the value of n is, we only need at most three bit-by-bit XOR operations to restore a pixel. Notably, each XOR is between a pair of 24-bit values if the image is color.

To observe this, assume that the size of secret image A is $w \times h$, then the number of XOR operations needed to evaluate $C^* = C_{11} \oplus C_{21} \oplus \cdots \oplus C_{m1}$ is $(m-1) \times [(w \times h)/m] < w \times h$ because each C_{i1} has size $[(w \times h)/m]$. Then, to get $C_{12}, C_{22}, \ldots, C_{m2}, m \times [(w \times h)/m] = w \times h$ XOR operations are required to evaluate $C_{i2} = C_{i3} \oplus C^*$, here $1 \leqq i \leqq m$. Finally, to reveal A, $w \times h$ XOR operations are needed to evaluate $A = B_1 \oplus B_2$. Together, $[(3 \times m - 1)/m] \times (w \times h) < 3 \times (w \times h)$ XOR operations are required to reveal A from any k final shadows. On average, since image A has $w \times h$ pixels, at most three XOR operations are required to restore each pixel. Table 1 below shows a comparison with reported schemes. Obviously, the proposed scheme has the smallest decryption load on average. Notably, the

Table 1. Time complexity for decoding. (The time to reconstruct a pixel of image A.)

Schemes	(k, n) Threshold	(n, n) Threshold
Thien and Lin's polynomial scheme Ref. 9	$O(\log^2 k)$ (Math operations$^{(1)}$)	$O(\log^2 n)$ (Math operations)
Wang and Sue's polynomial scheme Ref. 11	$O(\log^2 k)$ (Math operations)	$O(\log^2 n)$ (Math operations)
Digitalized version of Refs. 3, 4, 15 and 16	$O(k \times \text{per}^{(2)})$ (OR operations)	$O(n \times \text{per})$ (OR operations)
Lukac and Plataniotis's schemes Refs. 5–8	$O(k \times \text{per})$ (OR-like operations) for Refs. 5–7. Reference 8 is for (2, 2) case only; there is no (k, n) case in Ref. 8.	$O(n \times \text{per})$ (OR-like operations) for Refs. 5–7. Reference 8 is for (2, 2) case only, and it needs only $2 - 1 = 1$ OR-like operation.
Fang and Lin's scheme Ref. 2	$O(k \times \text{per})$ (OR operations)	$n - 1$ (XOR operations)
Wang *et al.*'s scheme Refs. 12 and 17	$O(k \times \text{per})$ (OR operations) in Ref. 17. Reference 12 gave no (k, n) scheme$^{(3)}$ unless $k = 2$; and its $(2, n)$ scheme uses only $2 - 1 = 1$ XOR operation to reconstruct a pixel of A.	$O(n \times \text{per})$ (OR operations) in Ref. 17. $n - 1$ (XOR operations) in Ref. 12.
Our scheme	3 (XOR operations)	3 (XOR operations)

(1) Math operations: $+, -, \times, \div$.
(2) Note that per means "Pixel expansion rate". Usually, per is a positive integer at least two in Refs. 2–7 and 15–17.
(3) When $k = 2$, the $(2, n)$ scheme in Ref. 12 is a very fast one, since only one XOR operation is needed. But their decoding is not loss-free.

proposed scheme also needs at most three XOR operations in encoding process to share each pixel of secret image into n final shadows $D_1, D_2, \ldots, D_n$, because the decoding process is exactly an inversion of an encoding one. Besides Table 1, the readers can also refer to Figs. 6 and 7 to see that our decoding time does not increase as n increases its value. Since, besides our method, Wang's[12] is one of the fastest schemes in Table 1, we only compare our CPU time with Ref. 12 in Fig. 6. As for Fig. 7, because Ref. 12 has no (k, n) design if $2 < k < n$, no curve for Ref. 12 is drawn there. (We only use this figure to show that our CPU time is really a constant.)

As for the space complexity, Thien and Lin's scheme[9] has a pixel expansion rate per $= 1/k$ for the (k, n) threshold cases. Wang and Su proposed the scheme[11] to reduce 40% of Thien and Lin's shadow images size. On the other hand, as the value of n increases, per is very large for digital versions of schemes.[3,4,15,16] Although the probabilistic scheme[14] has per $= 1$, the reconstructed secret image is not error-free. The per in Lukac and Plataniotis's schemes[5–7] are at least two. Lukac and Plataniotis's special method[8] has no pixel expansion problem (per $= 1$), but it is only for (2, 2) scheme. Although Fang and Lin's (n, n) and (k, n) schemes[2] have shadows of size smaller than Lukac and Plataniotis's,[7] their (k, n) scheme still has a per larger than one. The per in Wang *et al.*'s colored probabilistic (k, n) scheme[17] is still not less than one (per $\geqq 1$). As for Wang *et al.*'s deterministic (n, n) scheme,[12] per is one; but Ref. 12 does not have (k, n) schemes unless $k = 2$.

In the proposed scheme, our per is between 0 and 2; moreover, close to 0 is possible. To see this, let the size of secret image A be $w \times h$. Since the size of every temporary shadow C_i ($1 \leqq i \leqq m$) is $2\times\ (w \times h)/m$, the size of every final shadow D_i ($1 \leqq i \leqq n$) is

$$\begin{aligned}\left\lceil \frac{2\times(w\times h)}{m} \right\rceil \times C_{k-1}^{n-1} &= \left\lceil \frac{2\times(w\times h)}{C_{k-1}^{n}} \right\rceil \times C_{k-1}^{n-1} \\ &= \frac{2(w\times h)(n-k+1)}{n}.\end{aligned}$$

Here, we have used the fact that each final shadow D_i contains C_{k-1}^{n-1} temporary shadows. Now, after dividing the above by the size of A, we get our pixel expansion rate, i.e.

$$\text{per} = 2 \times \frac{(n-k+1)}{n} < 2, \quad \text{true for any } (k, n). \tag{4}$$

Therefore, each final shadow will be at most two times larger than secret image A. When n is very large and k is two, the rate converges to its upper bound 2. On the other hand,

$$\text{per} < 1 \quad \text{if } k > \frac{1+n}{2}. \tag{5}$$

In the special case when $k = n$, our per is $2/n$, and hence,

$$\text{per} = \frac{2}{n} \to 0 \quad \text{if } k = n \to \infty. \tag{6}$$

Therefore, each shadow will be very small when $k = n$. (See Fig. 5, for example, in which n was only 4, so per $= 2/n = 2/4 = 0.5$. If we had used a very large n, then per would have been much smaller.)

In summary, the proposed scheme does not have a serious pixel expansion problem or huge storage-space demanding for shadows (see Table 2).

4.3. *Perfect reconstruction and complications in implementation*

In Table 3, we provide the information about perfect reconstruction. Most schemes mentioned here are lossless in recovery, including our scheme. Exceptions are digitalized versions of Refs. 3, 4, 17, and the (2, n) scheme of Ref. 12 when $2 < n$.

Finally, the information about the level of easy-implementation is provided in Table 4. In summary,[1,9–11,13] evaluated polynomials and the remaining references used OR-like or XOR or look-up tables.

Table 2. Comparison of the pixel expansion rate (per) when shadows are created.

Schemes	(k, n) Threshold	(n, n) Threshold
Thien and Lin's polynomial scheme Ref. 9	$1/k$	$1/n$
Wang and Sue's polynomial scheme Ref. 11	$(1/k) \times 60\%$	$(1/n) \times 60\%$
Digital versions of Refs. 3, 4, 15 and 16	per is at least 2	per is at least 2
Lukac and Plataniotis's schemes[5–8]	per is at least 2. (per = 1 in Ref. 8, but Ref. 8 is for (2, 2) case only.)	per is at least 2. (per = 1 in Ref. 8, but Ref. 8 is for (2, 2) case only.)
Fang and Lin's scheme Ref. 2	$m \times n/(n+1)$ for some integer $m \geqq 2$.	$n/(n+1)$
Wang *et al.*'s scheme Refs. 12 and 17	per $\geqq 1$ in Ref. 17. (Reference 12 gave no (k, n) scheme unless $k = 2$; and per = 1 in its $(2, n)$ scheme.)	per $\geqq 1$ in Ref. 17. (per = 1 in (n, n) scheme Ref. 12.)
Our scheme	$0 <$ per $= 2 \times (n - k + 1)/n < 2$	$0 <$ per $= 2/n \leqq 1$

Table 3. Comparison of the perfect reconstruction ability.

Schemes	(k, n) and (n, n)
Thien and Lin's polynomial scheme[9]	Lossless recovery
Wang and Sue's polynomial scheme[11]	Lossless recovery
Digitalized versions of Refs. 3, 4, 15 and 16	Refs. 15 and 16 are lossless, but Refs. 3 and 4 cause loss.
Lukac and Plataniotis's schemes[5–8]	Lossless recovery
Fang and Lin's[2]	Lossless recovery
Wang *et al.*'s scheme[12,17]	Recovery might be lossy in Ref. 17 if per is close to 1. The $(2, n)$ scheme of Ref. 12 causes loss (Reference 12 gives no (k, n) scheme unless $k = 2$.) The (n, n) scheme of Ref. 12 is lossless.
Our scheme	Lossless recovery

5. Experimental Results

In our experiment, the input color image A is the popular test-image shown in Fig. 3(a). For $(k, n) = (2, 4)$ case, $n = 4$ final shadows D_1, D_2, D_3, D_4 generated in Sec. 3.1 are shown in Figs. 3(b)–3(e), and each has a size $2 \times (n - k + 1)/n = 2 \times (4 - 2 + 1)/4 = 3/2$ times larger than size of A. Figure 3(f) shows the error-free recovered A using any $k = 2$ of the four final shadows.

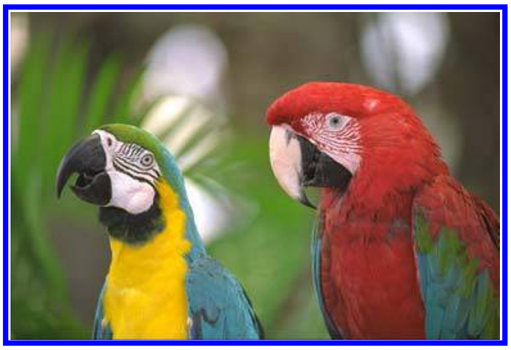
Other experiments dealing with $(k, n) = (3, 4)$ and $(k, n) = (4, 4)$ cases are shown in Figs. 4 and 5 respectively. And their pixel expansion rates are $2 \times (n - k + 1)/n = 2 \times (4 - 3 + 1)/4 = 1$ and $2 \times (n - k + 1)/n = 2 \times (4 - 4 + 1)/4 = 1/2$, respectively.

To show our constant decoding-time property, we also record in Figs. 6 and 7 the actual CPU time taken in decoding. The computer used is an IBM laptop with an Intel Pentium 1.70 GHz CPU, and the operating system is Microsoft Window

Table 4. The complicatedness (core techniques used) in implementation.

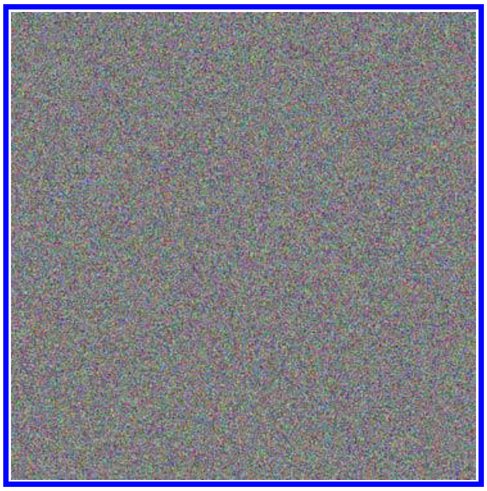
Schemes	Encoding Implementation	Decoding Implementation
Thien and Lin's polynomial scheme Ref. 9	Evaluate a polynomial	Use Largrange's interpolation
Wang and Sue's polynomial scheme Ref. 11	Evaluate a polynomial	Use Largrange's interpolation
Digitalized versions of Refs. 3, 4, 15 and 16	Either look up the basis matrices, or use OR operations.	Either look up the basis matrices, or use OR operations.
Lukac and Plataniotis's schemes Refs. 5–8	Either look up the basis matrices, or use OR-like operations.	Either look up the basis matrices, or use OR-like operations.
Fang and Lin's scheme Ref. 2	Either look up the basis matrices, or use OR-like operations.	Use XOR operations
Wang *et al.*'s scheme Refs. 12 and 17	Reference 17 either look up the basis matrices, or use OR operations. Reference 12 uses {AND, XOR} operations in the first scheme; uses XOR operations in the second scheme.	Reference 17 either look up the basis matrices, or use OR operations. Reference 12 uses XOR operations in both first and second schemes.
Our scheme.	Use XOR operations	Use XOR operations

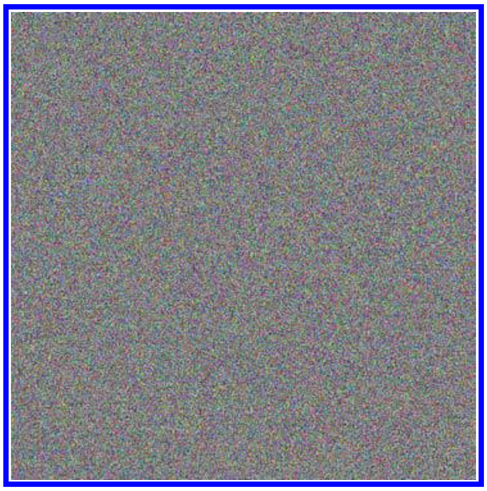
XP SP2. From Fig. 6, which deals with (n, n) system, it can be seen that our decoding time really does not vary as the value of n varies, but this is not the case for Wang *et al's* scheme.[12] Notably, for all (k, n) systems, our decoding time still remains constant as n increases its value. An example showing this is given in Fig. 7 in which $k = n/2$. Note that Wang *et al.*'s[12] does not have (k, n) systems unless k is n or 2.

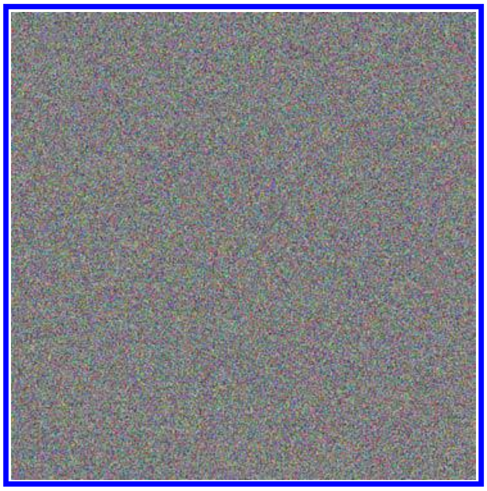


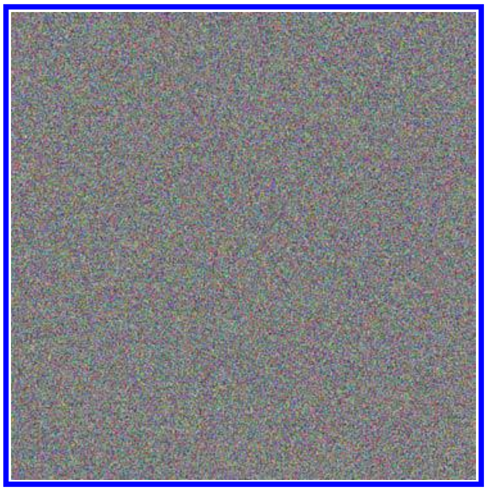
(a)

Fig. 3. An example of $(k = 2, n = 4)$. Here, (a) is the given 24-bit-per-pixel color image A; (b–e) are our final shadows D_1, D_2, D_3, D_4; (f) is the recovered error-free A using any two of the four final shadows.


(b)


(c)


(d)


(e)

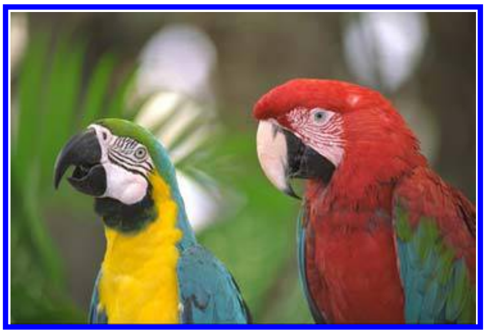

(f)

Fig. 3. (*Continued*)

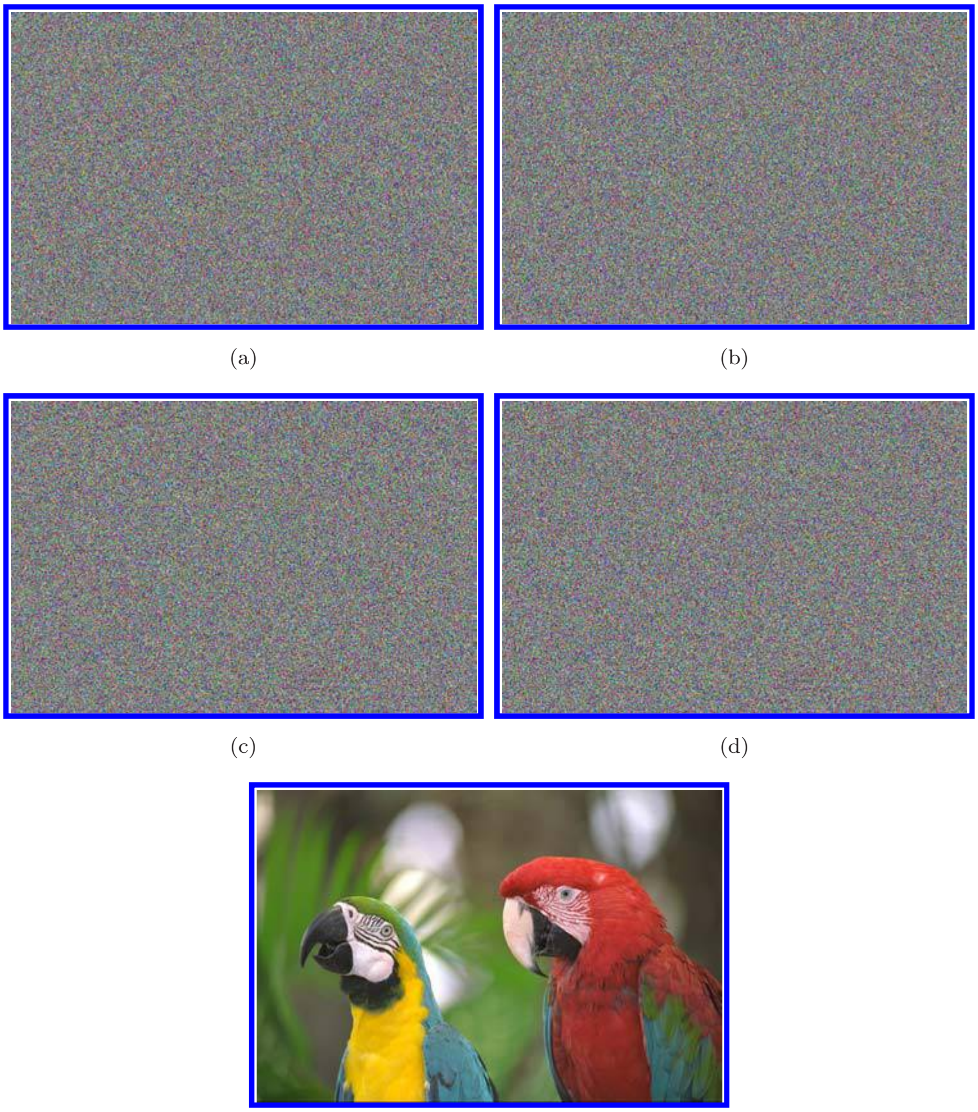
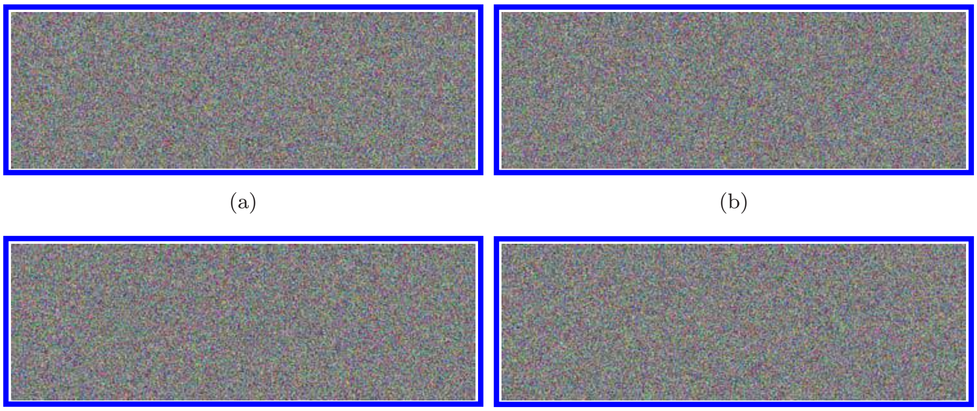


Fig. 4. An example of ($k = 3$, $n = 4$). Here, (a–d) are our final shadows D_1, D_2, D_3, D_4; (e) is the recovered error-free A using any three of the four final shadows.

6. Conclusion

In polynomial-based sharing approach, the shadow size is never a problem, but the decoding speed is very slow due to the polynomial-interpolation evaluation. To the contrary, storage space for shadows is large for almost all fast methods (the pixel expansion rate per is usually at least 2 for (k, n)-threshold schemes, and per $= 1$ is limited to loss causing schemes or some (n, n) nonthreshold schemes). In this


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

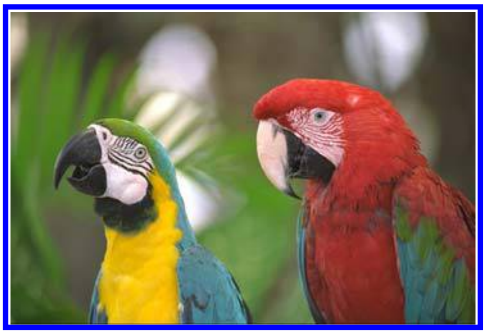

(e)

Fig. 5. An example of ($k = 4$, $n = 4$). Here, (a–d) are our final shadows D_1, D_2, D_3, D_4; (e) is the recovered error-free A using all four final shadows.

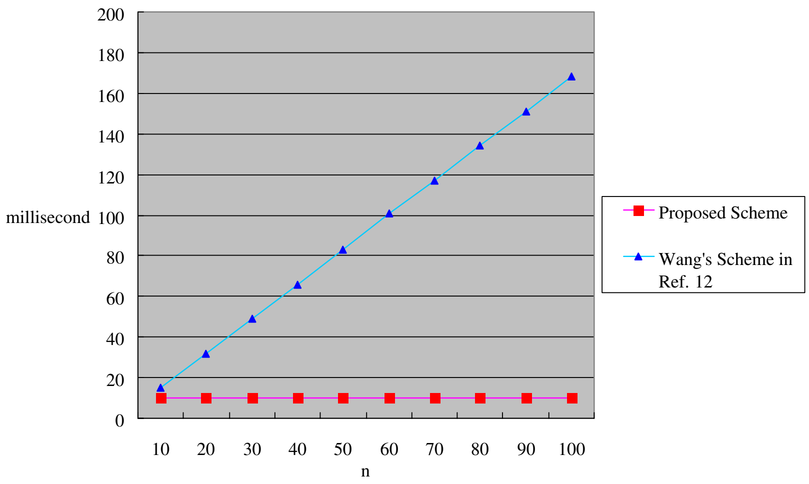


Fig. 6. The CPU time (milliseconds) for decoding (n, n) systems.

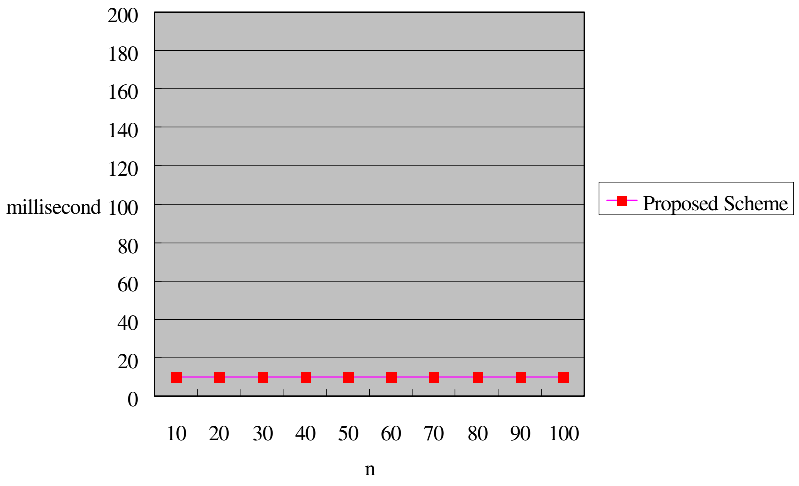


Fig. 7. The CPU time (milliseconds) for decoding each ($n/2$, n) systems by our scheme. There is no curve for Wang *et al.*'s scheme,[12] for their scheme has no ($n/2$, n) system or other (k, n) systems when $2 \leqq k < n$.

paper, we have designed successfully a scheme so that: (1) the generated shadows are of reasonable size (per is between 0 and 2; and close to 0 when the k is large and close to n [see Eqs. (4)–(6)], e.g. per $= 2/n$ in all (n, n) schemes); (2) the scheme only needs three 24-bit XOR operations per pixel to get a recovery of the given color image; and (3) unlike some probabilistic approaches, our recovered images are lossless; (4) our scheme is missing-allowable because it is a (k, n)-threshold scheme which requires only k out of n shadows that appear in the recovery meeting.

We have implemented the cases with (k, n) being, respectively, (2, 2), (2, 3), (3, 3), (2, 4), (3, 4), (4, 4), etc. The results are satisfactory in terms of the above advantages. Notably, the method also works for binary or grayscale image because the method is based on bit-by-bit operation. In fact, the given secret image A can be B-bit per pixel for any positive integer B (for example, use $B = 1$ for binary image, $B = 7$ or 8 for grayscale image, $B = 15$ for 5-5-5 pseudo color image, $B = 24$ for 8-8-8 color image).

Acknowledgments

This work is supported by National Science Council, Taiwan, R.O.C., under grant NSC 94-2213-E-009-093. The authors thank the reviewers and the editor for valuable suggestions.

References

1. S. K. Chen and J. C. Lin, Fault-tolerant and progressive transmission of images, *Patt. Recogn.* **38** (2005) 2466–2471.

2. W. P. Fang and J. C. Lin, Multi-channel secret image transmission with fast decoding: by using bit-level sharing and economic-size shares, *IJCSNS — Int. J. Comput. Sci. Network Security* **6** (5B) (2006) 228–234.
3. Y. C. Hou, Visual cryptography for color images, *Patt. Recogn.* **36** (2003) 1619–1629.
4. C. C. Lin and W. H. Tsai, Visual cryptography for gray-level images by dithering techniques, *Patt. Recogn. Lett.* **24** (2003) 349–358.
5. R. Lukac and K. N. Plataniotis, A color image secret sharing scheme satisfying the perfect reconstruction property, *IEEE 6th Workshop on Multimedia Signal Processing* (2004), pp. 351–354.
6. R. Lukac and K. N. Plataniotis, Color image secret sharing, *IEE Electron. Lett.* **40** (2004) 529–531.
7. R. Lukac and K. N. Plataniotis, Bit-level based secret sharing for image encryption, *Patt. Recogn.* **38** (2005) 767–772.
8. R. Lukac and K. N. Plataniotis, A cost-effective encryption scheme for color images, *Real-Time Imag.* **11** (2005) 454–464.
9. C. C. Thien and J. C. Lin, Secret image sharing, *Comput. Graph.* **26**(5) (2002) 765–770.
10. C. C. Thien and J. C. Lin, An image-sharing method with user-friendly shadow images, *IEEE Trans. Circuits Syst. Vid. Technol.* **13**(12) (2003) 1161–1169.
11. R. Z. Wang and C. H. Su, Secret image sharing with smaller shadow images, *Patt. Recogn. Lett.* **27** (2006) 551–555.
12. D. Wang, L. Zhang, N. Ma and X. Li, Two secret sharing schemes based on Boolean operations, *Patt. Recogn.* **40** (2007) 2776–2785.
13. Y. S. Wu, C. C. Thien and J. C. Lin, Sharing and hiding secret images with size constraint, *Patt. Recogn.* **37** (2004) 1377–1385.
14. C. N. Yang, New visual secret sharing schemes using probabilistic method, *Patt. Recogn. Lett.* **25** (2004) 481–494.
15. C. N. Yang and T. S. Chen, Aspect ratio invariant visual secret sharing schemes with minimum pixel expansion, *Patt. Recogn. Lett.* **26** (2005) 193–206.
16. C. N. Yang and T. S. Chen, Extended visual secret sharing schemes: improving the shadow image quality, *Int. J. Patt. Recogn. Artif. Intell.* **21** (2007) 879–898.
17. F. Yi, D. Wang, X. Li and Y. Dai, Colored probabilistic visual cryptography scheme with reversing, *Security and Management* (2007) 138–141.

Kun-Yuan Chao received his B.S. degree in computer and information science in 1996 from National Chiao Tung University, Taiwan. Then he received his M.S. degree in computer science and information engineering in 1999 from National Taiwan University, Taiwan. He is currently a PhD candidate of computer and information science at the National Chiao Tung University, Taiwan.

His recent research interests include secret image sharing, visual cryptography and image processing.

Ja-Chen Lin received his B.S. degree in computer science and M.S. degree in applied mathematics, both from National Chiao Tung University, Taiwan. He received his PhD degree in mathematics from Purdue University, USA. He joined the Department of Computer and Information Science at National Chiao Tung University in 1988, and then became a professor there.

His research interests include pattern recognition and image processing.

This article has been cited by:

1. Sachin Kumar, Rajendra K. Sharma. 2014. Threshold visual secret sharing based on Boolean operations. *Security and Communication Networks* **7**:3, 653-664. [CrossRef]
2. Ahmed A. Abd El-Latif, Xuehu Yan, Li Li, Ning Wang, Jia-Liang Peng, Xiamu Niu. 2013. A new meaningful secret sharing scheme based on random grids, error diffusion and chaotic encryption. *Optics & Laser Technology* **54**, 389-400. [CrossRef]
3. Lin Dong, DaoShun Wang, ShunDong Li, YiQi Dai. 2012. (2,n) secret sharing scheme for gray and color images based on Boolean operation. *Science China Information Sciences* **55**:5, 1151-1161. [CrossRef]
4. Tapasi Bhattacharjee, Jyoti Prakash Singh, Amitava Nag. 2012. A Novel (2,n) Secret Image Sharing Scheme. *Procedia Technology* **4**, 619-623. [CrossRef]
5. CHIN-CHEN CHANG, KUO-NAN CHEN, NGOC-TU HUYNH. 2011. LOSSLESS SECRET SHARING SCHEME WITH HIGH QUALITY SHARES BASED ON VQ-COMPRESSED IMAGES. *International Journal of Pattern Recognition and Artificial Intelligence* **25**:04, 529-546. [Abstract] [References] [PDF] [PDF Plus]