Multi-channel Secret Image Transmission with Fast Decoding: by using Bit-level Sharing and Economic-size Shares

Wen-Pinn Fang, and Ja-Chen Lin,

Department of Computer Science, National Chiao Tung University, Hsinchu 300, Taiwan.

Summary
The paper proposes a method to distribute images over network. There are several good properties: safety, lossless reconstruction, and real-time decoding. The size of each transmitted share is also competitive, as compared with an earlier work which is also bit-level based..

Key words:
Data hiding, image sharing; real-time decoding; share-size; lossless

Introduction

To transmit or store an image in a safer way against interceptor, there are at least three possible major approaches: encryption with keys [1]; hiding the image in other media or objects [2-3]; sharing the image among distinct channels/places [4-17]. Mixing these three major branches are also possible. In this paper, we focus on image secret sharing. In an (r,n) image sharing system ($r \le n$), the given image is encrypted into n shadow, noisy-like, images called shares The generated shares can be independently transmitted through various communication channels. To recover the image, at least r shares should be accessible at the same time. As a result, sharing among n of the many existing channels can balance between fault-tolerance (up to n-r used channels can be disconnected) and security (up to r-1 of the n used channels can be intercepted, not to mention that each of the much-more-than-n channels is usually filled with many other coy or ordinary images not related to the given important image).

To share an image, a possible way is to use polynomial-style sharing (PSS) (see Ref. [5-7]). Using PSS can get shares of smaller size, but it is "extremely" time-consuming in the decoding phase to recover the image from the r received shares. There is another way to share an image, as stated below. In [11-14] are non-expansive secret sharing scheme which can be used as private-key cryptosystem. In [15-17] can also recover the secret image in its original quality. In Ref. [8], Lukac and Plataniotis successfully applied visual-cryptography (VC) techniques in a bit-level manner (using VC on each bit-planes) and obtain another type of image sharing method having the following good properties:

1. real-time decoding, as opposed to PSS approach;

2. lossless recovery of images (PSS is also lossless).

If people really have to pick an disadvantage of this method in [8], then maybe the only disadvantage is that each (grey or colour) share is several times bigger than the given image. In the current paper, we propose an alternative method that is also bit-level based. This new method keeps the above advantages of [8], and uses shares of size smaller than that of [8] (thus reduce their transmission time and storage space). In the (r = n) case, our share-size is even smaller than the size of the input image; while the share-size in [8] is, say, 4 times greater the input image. (As for the (r,n) case (with $r<n$), our share-size is a little smaller than that in [8].)

The rest of this paper is organized as follows. The method is stated in Sec. 2; which contains two subsections: Sec. 2.1 is for the (n,n) case, whereas Sec. 2.2. is for the (r,n) case. Experimental results are shown in Sec. 3. Discussion is in Sec. 4.

2. The Proposed method

For a given H×W gray-scale or color image G, and for a given pair of parameters (r,n), our method to create the n expected shares is as follows (also see Fig. 1):

1 Physically split the secret image G, whose size is H×W, into two parts: upper parts and lower parts (see Fig. 2). For the given parameter pair (r, n), the upper part consists of first [n/(n+1)]×H×W pixels of the input image.

2 Notably, there are 8 bit-planes for a gray image (or, 24 bit-planes if color). Each bit-plane $B = B_U \cup B_L$ also has its own upper part B_U and lower part B_L. The upper part B_U is the first [n/(n+1)]×H×W bits of plane B; while the lower part B_L is the remaining [1/(n+1)]×H×W bits.

3. Sequentially pick a not-yet-processed bit-plane $B = B_U \cup B_L$ to process, until all 8 (or 24) bit-planes are processed. Each bit-plane is processed by two sub-steps to generate n binary-value shares {S1,…,Sn} for this bit-plane. The two sub-steps are:

Manuscript received May 5, 2006.
Manuscript revised May 25, 2006.

(3-i) Share the lower part.

(3-ii) Then, share the upper part.

(The details of (3-i) and (3-ii) are described later in Sec. 2.1 for the case r=n;and then described again in Sec. 2.2 for the case r<n.)

4. The generated share bits are used to produce gray-scale(or color) share images suitable for secure distribution over untrusted networks.

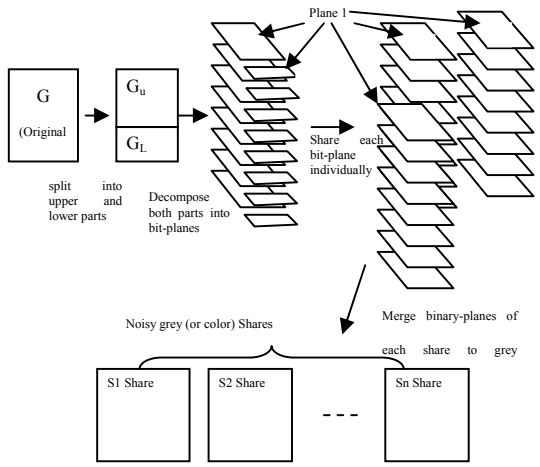


Fig .1 Flowchart of the proposed method.

Fig. 2. An example of split. (a) is the original, (b1) is the upper part, and (b2) is the lower part.

2.1 Sharing a bit plane B in the (n,n) case.

Step 1. (Sharing the Lower part (B_L) of B.)

For each share Sm, where m =1,2,…,n-1, randomly assign its bits located at {ni+m: i=0,1,2,….}. As for the last share, i.e. Share Sn , we do not randomly assign its bits located at {ni+n; i=0,1,2,….}. Instead, we use the lower part (B_L) of B to compute the bit-values of the share Sn at location {ni+n; i=0,1,2,….}. More specifically, for each i=0,1,2, …, we compute the value of Sn(ni+n) by

$Sn(ni+n) = [S1(ni+1) \oplus S2(ni+2) \oplus \ldots \oplus Sn-1(ni+[n-1])] \oplus B_L(i)$

where $\oplus$ is the exclusive-OR operator. Note that Sn (ni+n) is computed this way because later we can recover the lower part of B by using

$B_L(i) = S1(ni+1) \oplus S2(ni+2) \oplus \ldots \oplus Sn-1(ni+[n-1]) \oplus Sn(ni+n)$

Step 2. (Sharing the Upper part (B_U) of B.)

Use data B_U (the upper part of B) to determine the remaining bits of all shares. The requirement is very simple: at each position t, we require that

$B_U(t) = S1(t) \oplus S2(t) \oplus \ldots \oplus Sn(t)$.

In other words, we only requires that: there are "odd" number of 1s appearing in the n-bits set {S1 (t), S2 (t), …, Sn (t)} if and only if $B_U(t)=1$. ♦♦

Example (k=3,n=3)

Assume the upper n/(n+1)=3/(3+1)=3/4 part of the original bit-plane B is B_U= 1010101…; and assume the lower 1/(n+1)=1/(3+1)=1/4 part of B is B_L=1110…. Then, we show how to use the above algorithm to produce the n=3 shares {S1, S2, S3}, where each share has (3H/4)×W bits when the bit-plane B has H×W bits.

Step 1. For Share S1, randomly assign its bits at {1,4,7,…} = {3i+1: i=0,1,2,….}. For Share S2, randomly assign its bits at {2,5,8,…} = {3i+2: i=0,1,2,….}. However, for the last share, i.e. Share S3 (because n=3), we do not randomly assign its bits at {3,6,9,…} = {3i+3: i=0,1,2,….}. Instead, we use the lower part (B_L) of B to compute the values of these bits at location {3,6,9,…} of Share 3. More specifically, for each i=0,1,2,3,4…, we require that

$S3(3i+3) = S1(3i+1) \oplus S2(3i+2) \oplus B_L(i)$

for the purpose that later we can recover the lower part of B by using

$B_L(i) = S1(3i+1) \oplus S2(3i+2) \oplus S3(3i+3)$.

For example, since BL is assumed to be 1110... in this example, the above idea of Step 1 can be illustrated by Fig. 3.

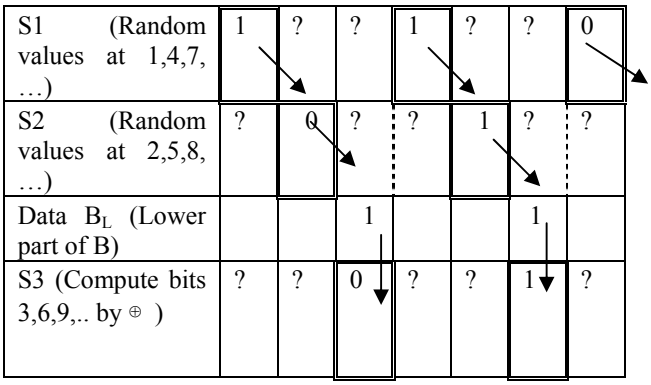


Fig. 3. Step 1 of the (k=3,n=3) example.

Step 2. Use data B_U (the upper part of B) to determine the remaining bits of all shares. The requirement is very simple: at each position t, we require that
$B_U(t) = S1(t) \oplus S2(t) \oplus S3(t)$.
In other words, there are "odd" number of 1s appearing in these three bits {S1(t), S2(t), S3(t)} if and only if $B_U(t)=1$. One of the many solutions is shown in Fig. 4. Note that all three shares have been created after Step 2.

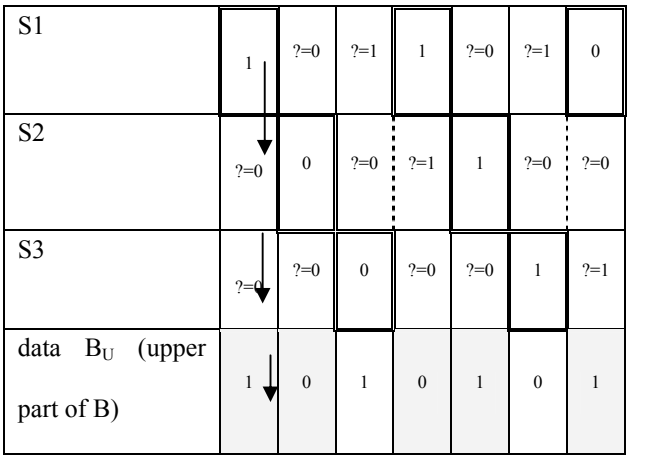


Fig. 4. Step 2 of the (k=3,n=3) example.

2.2 Sharing a bit plane B in the (r,n) case, i.e. when r<n

Step 0: Create a pair of basis matrices C^0 and C^1 for the (r,n) system. (The creation of C^0 and C^1 is introduced in many other papers, see Ref. [9] for example, we do not introduce the detail here.) No matter how they are created, this pair must have the following properties:

- Each matrix has n rows. Each entry of the two matrices is just a single bit whose value is either 0 or 1; each 1 means a black dot while each 0 means a white dot.
- "Stacking" r of the n rows of C^0 (or C^1) is defined as getting a row whose ith element is the result of using the "OR" operator (not "exclusive-OR") on the ith elements of the corresponding r rows.
- Stacking any r rows of C^0 together always get a row whose number of 1s are less than the number of 1s obtained from stacking any r rows of C^1.

Step 1 (To share the information of B_L, i.e. the lower part of B).
(1-i) Initially, let q=1.
(1-ii) Let p be the q-th bit of the string B_L, i.e. $p=B_L(q)$.
(1-iii) For m=1,2,…,n, use the m-th row of C^p to paint the ([q-1]n+m)-th block of the share Sm (m=1,2,…,n).
(1-iv). If q reaches $\frac{1}{n+1}$H, then Step 1 ends here, and we go to Step 2. Else, q ←q+1and go to (1-ii).
Step 2 (To share the information of B_U, i.e. the upper part of B).
(2-i) Initially, let q=1.
(2-ii) Let p be the q-th bit of the string B_U. Also, let m=q (mod n)
(2-iii-A) If the mth row of C^p is identical to the qth block of the share Sm, then, for all k=1,…,n, paint the qth block of the share Sk using the kth row of C^p. Then go to Step (2-iv).
(2-iii-B) If the mth row of C^p is different from the qth block of the share Sm, then, we need to permute the columns of C^p to get a temporary matrix $C^{p\prime}$ whose mth row is identical to the qth block of the share Sm. (This permutation subroutine is quite easy to design, so we omit it here.) Then, for all k=1,…,n, paint the qth block of the share Sk using the kth row of $C^{p\prime}$.Then go to Step (2-iv).
(2-iv). If q reaches $\frac{n}{n+1}$H, then go to post-processing. Else, let q ←q+1and go to (2-ii).

POST-PROCESSING: We already have n shares, and each share is a sequence of $\frac{n}{n+1}$H×W blocks. Now, convert each sequence of blocks from 1-dim block-string to its 2-dim image version. In other words, for each share, divide its $\frac{n}{n+1}$H× W blocks into $\frac{n}{n+1}$H rows. (The first W blocks form the first row; the next W blocks form the second row; etc.)

♦♦

Example (k=2,n=6)
Assume the upper n/(n+1)=6/(6+1)=6/7 part of the original bit-plane B is BU= 1001…; and assume the lower 1/(n+1)=1/(6+1)=1/7 part of B is BL=1011…. Then, we show how to use the above algorithm to produce the n=6 shares {S1, …, S6}, where each share has (6H/7)×W "blocks" when the bit-plane B has H×W bits.

Step 0. Create a pair of basis matrices C^0 and C^1 for the (2,6) system. Since n=6, each matrix has 6 rows. The two matrices in Fig. 5 obviously meet the requirements stated in the Step 0 of the algorithm.

C^0 :

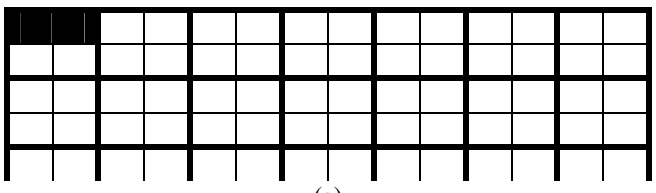
1	0	1	0
1	0	1	0
1	0	1	0
1	0	1	0
1	0	1	0
1	0	1	0

C^1 :

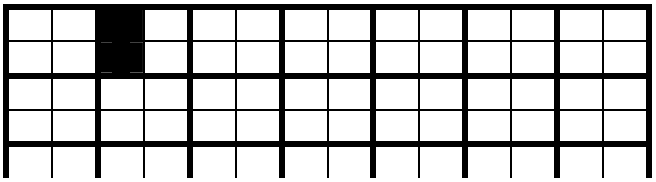
1	1	0	0
1	0	1	0
1	0	0	1
0	1	1	0
0	1	0	1
0	0	1	1

Fig. 5. The basis matrices C^0 and C^1 used for the (2,6) system in the example.

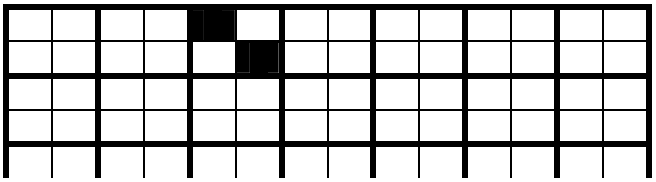
Step 1: (To share B_L=1011… , i.e. to share the lower part of B.)
Because the 1st bit of B_L is 1, We look up the matrix C^1. Then, use the 1st row of C^1 , i.e. use 1100 to paint 1st block of share S1.(Since 1100 means BBWW, we may use the first two entries (BB) to paint the upper half of the block, then use the next two entries (WW) to paint the lower half of the block.) Analogously, the share S2 uses 1010 (the 2nd row of C1) to paint its 2nd block. The share S3 uses 1001 (the 3rd row of C^1) to paint its 3rd block. The share S4 uses 0110 (the 4th row of C^1) to paint its 4th block. The share S5 uses 0101 (the 5th row of C^1) to paint its 5th block. The share S6 uses 0011 (the 6th row of C^1) to paint its 6th block. Then the cycle repeats itself again using next bit of B_L. Since $B_L(2)$=0, we use C^0 now. The rows 1-6 of C^0 are copied respectively (one row per share), to 7th block of S1, 8th block of S2, 9th block of S3, 10th block of S4, 11th block of S5, and 12th block of S6. This is Cycle 2. Third cycle uses BL(3)=1 to grab C^1 to paint the (12+i)ith block of the share Si (i=1,2,..6). The process repeats again and again until all bits of B_L are used.



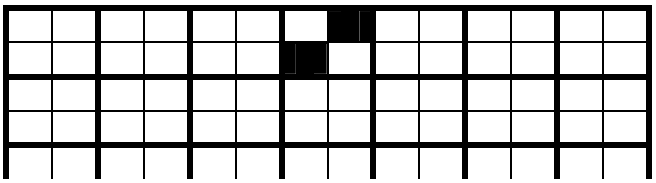
(a)



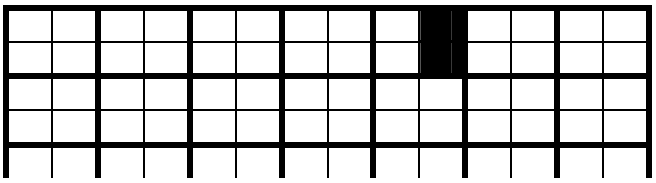
(b)



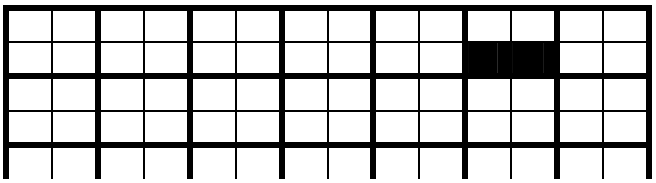
(c)



(d)



(e)

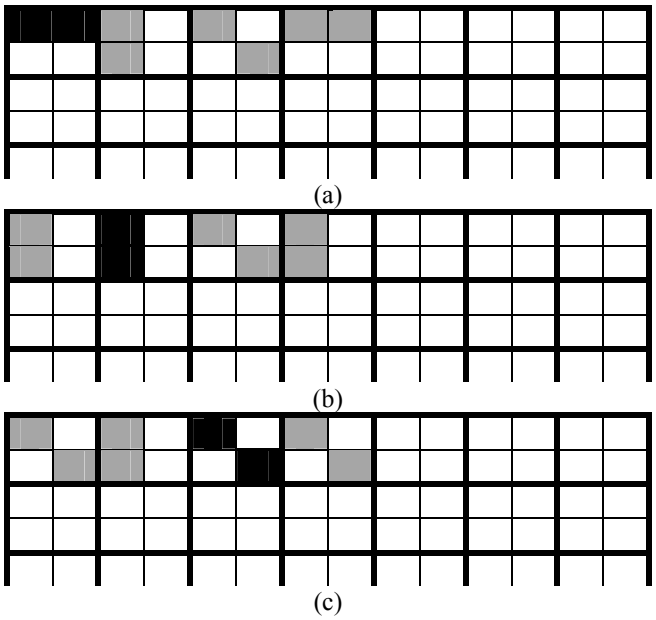


(f)

Fig. 6. Step 1 for the (r=2, n=6) example. Here, (a)-(f) are, respectively, the six shares S1 – S6.

Step 2: (To share B_U , i.e. share the upper part of B.)
Note that the upper part of B is 1001…; so, we look up first C^1, then C^0, then C^0 again, then C^1 again, and so on. First, because $B_U(1)$ = 1, we use the six rows of C^1 to paint the 1st blocks of shares S1-S6. In this painting, the painted pattern of the 1st block of the 1st Share (S1) has no contradiction with what it had been painted earlier in Step 1. (The block had been painted earlier in Step 1 as 1100 (i.e. BBWW), so, no contradiction if we use the 1st row of C^1 to paint it.) Therefore, the 1st blocks of all six shares

are done using the six rows of C^1. Now we proceed to $B_U(2)$. Since $B_U(2)=0$, we use the six rows of C^0 to paint the 2nd blocks of shares S1-S6. Again, this painting is accepted because the painted pattern of the 2nd block of the 2nd share (S2) has no contradiction with what it had been painted earlier in Step 1. Now we proceed to $B_U(3)$.Because $B_U(3)=0$, we also try to use the six rows of C^0 to paint the 3rd blocks for Shares S1-S6. However, we find that this will cause the 3rd block of the 3rd share (S3) has contradiction with what it is already painted in Step 1 earlier. (The block had been painted in Step 1 as 1001 (i.e. BWWB), rather than 1010 (i.e. BWBW)). Therefore, we permute the columns of C^0 to get a temporary matrix $C^{0\prime}$ whose 3rd row is also 1001. Then, we use the six rows of the new matrix $C^{0\prime}$ to paint the 3rd blocks of the six shares. Notably, all six rows of $C^{0\prime}$ become 1001 after this permutation of columns; that explains why the 3rd blocks of all six shares are painted as 1001 (BWWB) in Fig. 7. Now, we proceed to the 4th bit of B_U and find that $B_U(4)=1$, so we use the six rows of C^1 to paint the 4th blocks for Shares S1-S6. In the painting, the painted pattern of the 4th block of the 4th Share (S4) has no contradiction with what it had been painted in Step 1 earlier. (The block had been painted earlier in Step 1 as 0110 (i.e. WBBW), so, no contradiction if we use 4th row of C^1 to paint it.) Therefore, the 4th blocks of the six shares are done using the six rows of C^1. Our explanation ends here, for the remaining process are similar.



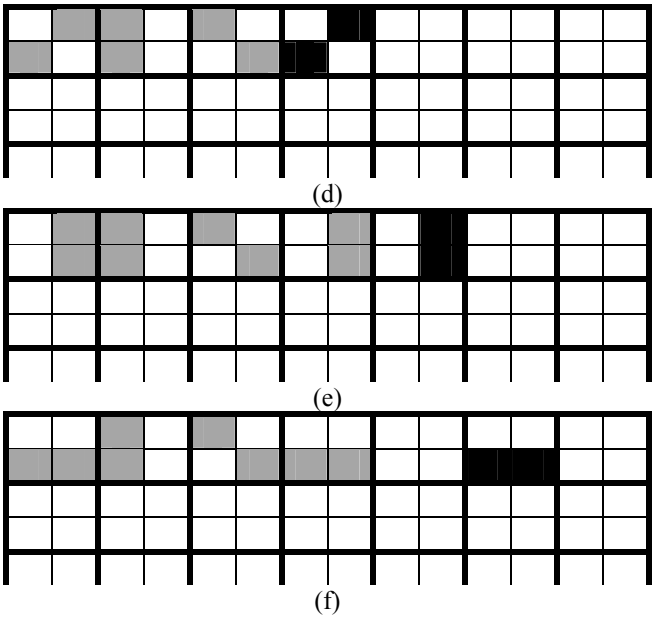


Fig. 7. Step 2 for the (r=2, n=6) example. Here, (a)-(f) are, respectively, the six shares S1 – S6. Darker elements were determined earlier in Step 1, and hence cannot be changed now in Step 2.

3. Experiments

In the first experiment, the (r,n) is (2,2). The result is shown in Fig 8, of which (a) is the input gray-value image Lena; (b) and (c) are the two gray-value shares (the size of each share is just $n/(n+1)=2/3$ of that of (a)); (d) is the restored error-free result using (b) and (c). The result can be compared with Ref. [8]. In Ref[8], their recovery is also error-free (just like ours), but each of their shares is 4 times lager than the input image, while each of our shares is $n/(n+1)=2/3$ times smaller than the input image. In other words, their share-size is $(2\times2)\times(3/2)=6$ times bigger than ours if the input image is the same.

In the second experiment, the (r,n) is (2,6). The result is shown in Fig 9, of which (a) is the input gray-scale image Lena; (b) is one of the six gray-value shares (each share is $(n/(n+1))\times(2\times2)= (6/7) \times(2\times2) =3.43$ times greater than (a)). Using any two of the six shares, we can get the error-free recovery of the input image (identical to (a)). The result can be compared with Ref. [8]. Their recovery is also error-free (just like ours), and each of their shares is 4 times lager than the input image, while each of our shares is $(2\times2) \times 6/7 =3.43$ times larger than the input image. Therefore, their share-size is $(n+1)/n = 7/6$ times bigger than ours if the input image is the same.

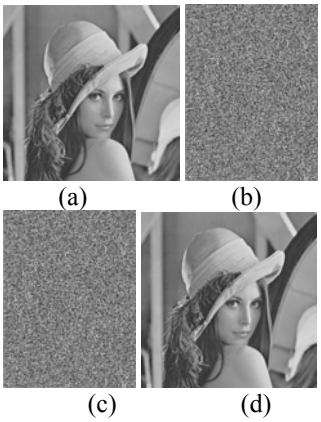


Fig. 8. An (r=2,n=2) experiment using our method. (a) is the input gray-value image; (b) and (c) are the two gray-value shares (the size of each share is just n/(n+1)=2/3 of that of (a)); (d) is the restored error-free result (identical to (a)) using (b) and (c).

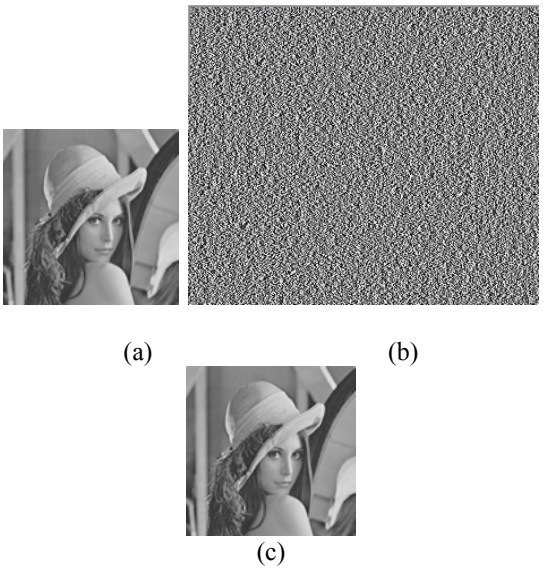


Fig. 9. An (r=2,n=6) experiment using our method. (a) is the input gray-value image; (b) is one of the six gray-value shares (each share is (n/(n+1))×(2×2)=(2×2)×6/7=3.43 times greater than (a)); (c) is the restored error-free result (identical to (a)) using any two of the six shares

(a) (b)

In general, in the (n,n) cases, e.g. the (2,2), or (3,3), or (4,4) cases, each of their shares is 2×2×(n+1)/n times larger than ours. Note that 2×2×(n+1)/n is a number between 4 and 6. We can therefore save more transmission time or storage space than the method in [8] does. On the other hand, in the (r,n) cases, then each of their shares is (n+1/n) times larger than ours. Note that (n+1)/n is a number between 1 and 4/3=1.33 (for n>2 in the (r,n) case, because r cannot be 1.) Of course, as compared with [8], we still have a little advantage of space-saving or communication-time-saving in the (r,n) cases, although the advantage is not as sharp as in the (n,n) cases.

When it is (r,n) case, our performance is similar to [PR2006], no matter it is encoding or decoding. But, when it is (n,n) case, our method is obviously faster than [8], no matter it is encoding or decoding. The reason is that, in the (n,n) case, we did not use any kind of blocks (for example, the 2×2 blocks) to expand any pixel. For readers' interest, we also list the processing time of the polynomial-style-sharing (PSS, see [TL02]). Its decoding time is definitely much slower than bit-level methods (ours and [8]). As for its encoding time, it beats us in the (r,n) case, but lose to us in the (n,n) case. The reason is that in the (n,n) case our system is extremely simple.

In summary, in the (n.n) case, we lead both [8] and PSS [5], no matter it is encoding or decoding. In the (r,n) case, PSS takes the lead in encoding, but falls far behind ours and [8] in decoding.

4. Discussion

In this paper, we have proposed a sharing method for grey or color images. The decoding speed, just like the one in [8], is real-time. The recovered image is lossless; so is [8]. But our method uses shares of size smaller than those used in [8], and hence, save transmission time and storage space. This advantage is particularly obvious in the (n,n) systems, i.e. when r=n. In that case, each grey/colour share is 4 to 6 times smaller than that used in [8].

Both [8] and our method are bit-level based; and each bit-plane is processed independently. Therefore, if the transmission time (or storage space) is too limited, people may discard some less important bit-planes. For example, discard the last 3 bit-planes and only use the most important 8-3=5 bit-planes, then each grey-value share is actually a physical-combination of five bit-plane shares rather than a combination of eight bit-plane shares. Therefore, each share is reduced in size to 5/8 of the original grey-value share. (If only the least important one of the eight bit-planes is removed, then the PSNR of the reconstructed image is about 52 db.)

Acknowledgments

This work is supported by National Science Council, Taiwan, R.O.C., under Grant NSC 94-2213-E-009-093.

References

[1] N. Bourbakis and A. Dollas, "SCAN-based compression-encryption-hiding for video on demand," IEEE Multimedia Magazine, Vol. 10, pp. 79-87, 2003.

[2] C.C. Thien and J.C. Lin ,"A simple and high-hiding capacity method for hiding digit-by-digit data in images based on modulus function", Pattern Recognition, Vol. 36 (12), pp. 2875-2881, 2003.

[3] C.K. Chan and L.M. Cheng, "Hiding data in images by simple LSB substitution", Pattern Recognition, Volume 37 (3), pp. 469-474, 2004.

[4] J.B. Feng, H.C. Wu, C.S. Tsai and Y.P. Chu, A new multi-secret images sharing scheme using Largrange's interpolation, Journal of Systems and Software, Volume 76 (3) , pp.327-339, 2005.

[5] C.C. Thien and J.C. Lin, "An Image-Sharing Method With User-Friendly Shadow Images," IEEE-Trans. on Circuits and Systems for Video Technology, Vol. 13, No. 12, pp. 1161-1169, 2003.

[6] C.C. Thien and J.C. Lin, "Secret image sharing," Computers and Graphics, Vol. 26, pp. 765-770, 2002.

[7] Y.S. Wu, C.C. Thien and J.C. Lin, "Sharing and hiding secret images with size constraint," Pattern Recognition, Vol. 37(7), pp. 1377-1385, 2004.

[8] R.Lukac and K.N. Plataniotis, "Bi-level based secret sharing for image encryption", Pattern Recognition, Vol. 38, Issue 5, pp. 767–772, 2005.

[9] M. Naor, A. Shamir, "Visual cryptography", Proc. Eurocrypt '94, Lecture Notes Computer Sci., Vol. 950, pp.1-12, 1994.

[10] W.P. Fang and J.C. Lin, "Visual cryptography with extra ability of hiding confidential data", J. Electronics Imaging, Vol.4, 2006.

[11] R. Lukac and K.N. Plataniotis, A cost-effective encryption scheme for color images. Real-Time Imaging, Special Issue on Multi-Dimensional Image Processing, vol.11, no.5-6, pp.454-464, October-December 2005.

[12] S. Sudharsan, "Shared key encryption of JPEG color images," IEEE Transactions on Consumer Electronics, vol. 51, no. 4, pp. 1204-1211, November 2005.

[13] R. Lukac and K.N. Plataniotis, "Digital image indexing using secret sharing schemes: a unified framework for single-sensor consumer electronics," IEEE Transactions on Consumer Electronics, vol. 51, no.3, pp.908-916, August 2005.

[14] R. Lukac and K.N. Plataniotis, "Colour image secret sharing," IEE Electronics Letters, vol. 40, no. 9, pp. 529-530, April 2004.

[15] D. Jin, W.Q. Yan, M. S. Kankanhalli, Progressive color visual cryptography, Journal of Electronic Imaging, vol. 14, no. 3, 033019, Jul-Sep. 2005.

[16] C. C. Lin and W. H. Tsai, "Secret image sharing with capability of share data reduction," Optical Engineering, vol. 42, pp. 2340-2345, August 2005.

[17] R. Lukac and K.N. Plataniotis, Image representation based secret sharing. Communications of the CCISA, Special Issue on Visual Secret Sharing, vol. 11, no. 2, pp. 103-114, April 2005.

Wen-Pinn Fang received his BS degree in mechanical engineering in 1994 from National Sun-Yet-Sen University and his MS degree in mechanical engineering in 1998 from National Chiao Tung University, where he is currently a PhD candidate in the Computer and Information Science Department. His recent research interests include pattern recognition and image processing..

Ja-Chen Lin *received his BS degree in computer science in 1977 and his MS degree in applied mathematics in 1979, both from National Chiao Tung University, Taiwan, and his PhD degree in mathematics from Purdue University, U.S.A., in 1988. In 1981 to 1982 he was an instructor with the National Chiao Tung University and from 1984 to 1988 he was a graduate instructor with Purdue University. In August 1988 he joined the Department of Computer and Information Science at National Chiao Tung University, where he is currently a professor. His recent research interests include pattern recognition and image processing. Dr. Lin is a member of the Phi- Tau-Phi Scholastic Honor Society.*