PERGAMON

Computers & Graphics 26 (2002) 765–770

COMPUTERS & GRAPHICS

www.elsevier.com/locate/cag

Technical section

# Secret image sharing ☆

Chih-Ching Thien, Ja-Chen Lin*

*Department of Computer and Information Science, National Chiao Tung University, Hsinchu 300, Taiwan, ROC*

**Abstract**

In this paper, we propose a method such that a secret image is shared by $n$ shadow images, and any $r$ shadow images ($r \leqslant n$) of them can be used to restore the whole secret image. The size of each shadow image is smaller than the secret image in our method. This property gives the benefit in further process of the shadow images, such as storage, transmission, or image hiding. 

*Keywords:* Secret sharing; Lossless reveal; Shadow images; Gray value range

## 1. Introduction

Secret texts and images often exist in the commercial or military application. About the storage of the secret data or images, the security is of big concern. In recent years, many techniques to increase the security of the secret were proposed; examples include image hiding and watermarking. However, a common weakness of these techniques is that the secret data are all in a single information-carrier, and the secret data cannot be revealed completely if the information-carrier is lost or crippled. If we use many duplicates to overcome the weakness, the danger of security exposure will also increase. To solve this dilemma, secret sharing method might be one of the possible solutions.

Blakley [1] and Shamir [2] first independently proposed the concept of secret sharing. It was called the $(r, n)$ threshold scheme. The succeeding studies were mainly related to the security of the keys [3–6]. When the secret data are in fact secret images, because the number of bytes used in a digital image is usually very large (for example, $512 \times 512$) and the gray value is bounded (0–255), using the $(r, n)$ threshold scheme directly will waste a lot of memory space; we should therefore develop a specific method for secret image sharing. In this paper, we propose a secret image sharing method derived from the $(r, n)$ threshold scheme. In our method, the size of each shadow image will be smaller than that of the secret image, as will be seen later. We will require that:

1. The secret image is used to generate $n$ shadow images.
2. Any $r$ or more shadow images can be used to reconstruct the secret image.
3. Any $r-1$ or less shadow images cannot get sufficient information to reveal the secret image.

In the rest of this paper, the $(r, n)$ threshold scheme is reviewed in Section 2. Section 3 introduces our secret sharing method. The experimental result is shown in Section 4. In Section 5, we provide the security analysis. The benefits of the size reduction of the shadow images are discussed in Section 6. Finally, the conclusions are stated in Section 7.

## 2. The $(r, n)$ threshold scheme: a review

Because our method is based on the $(r, n)$ threshold scheme proposed by Shamir, we roughly introduce his scheme first. Suppose that we want to divide the secret data $D$ into $n$ shadows ($D_1, \ldots, D_n$), and we wish that the secret data $D$ cannot be revealed without $r$ or more

☆ This work is supported by National Science Council, R.O.C under grant NSC90-2213-E-009-131.

*Corresponding author. Department of Computer and Information Science, National Chiao Tung University, 1001 Ta Hsueh Road, Hsinchu 30050, Taiwan, ROC. Tel.: +886-03-5715900; fax: +886-03-5721490.

*E-mail address:* jclin@cis.nctu.edu.tw (J.-C. Lin).

0097-8493/02/$ - see front matter 
PII: S0097-8493(02)00131-0

shadows. Without loss of generality, we can assume that the data $D$ is a number. To split $D$ into $n$ shadows, we randomly pick a prime number $p$ and an $r-1$ degree polynomial

$$q(x) = (a_0 + a_1 x + \cdots + a_{r-1}x_{r-1}) \bmod p, \quad (1)$$

in which $a_0 = D$, and then evaluate

$$D_1 = q(1), \ldots, D_i = q(i), \ldots, D_n = q(n). \quad (2)$$

Note that each $D_i$ is a shadow. The coefficients $a_1, \ldots, a_{r-1}$ in $q(x)$ are randomly chosen from the integers in $0-(p-1)$. Given any $r$ pairs of these $n$ pairs $\{(i, D_i)\}_{i=1}^{n}$, we can find the coefficients $a_0-a_{r-1}$ of $q(x)$ by the Largrange's interpolation, and hence the secret data $D = a_0$ is also revealed.

In image sharing, to use Shamir's $(r, n)$ threshold scheme directly, $a_0$ is taken as the gray value of the first pixel, then we obtain the corresponding output $q(1)-q(n)$; after that, $a_0$ is replaced by the gray value of the second pixel, and the process repeats until all pixels of the secret image are processed. Hence, each shadow image is also of size $512 \times 512$ if the secret image is $512 \times 512$.

## 3. The proposed secret image sharing method

### 3.1. The sharing phase

Suppose that we want to divide the secret image $D$ into $n$ shadow images $(D_1, \ldots, D_n)$, and the secret image $D$ cannot be revealed without $r$ or more shadow images. In the proposed method, we generate the $r-1$ degree polynomial, by letting the $r$ coefficients be the gray values of $r$ pixels. Therefore, the major difference between our method and Shamir's is that we use no random coefficient.

Because the gray value of a pixel is between 0 and 255, we let the prime number $p$ be 251 which is the greatest prime number not larger than 255. To apply the method, we must truncate all the gray values 251–255 of the secret image to 250 so that all gray values are in the range 0–250. (If we do not want to change the gray values of the secret image, there is a lossless secret image sharing method proposed later in Section 3.3.) The image is divided into several sections (explained later in Step 3). Each section has $r$ pixels, and each pixel of the image belongs to one and only one section. For each section $j$, we define the following $r-1$ degree polynomial

$$q_j(x) = (a_0 + a_1 x + \cdots + a_{r-1}x_{r-1}) \bmod 251, \quad (3)$$

where $a_0, \ldots, a_{r-1}$ are the $r$ pixels of the section, and then evaluate

$$q_j(1), q_j(2), \ldots, q_j(n). \quad (4)$$

The $n$ output pixels $q_j(1)-q_j(n)$ of this section $j$ are sequentially assigned to the $n$ shadow images. Since for each given section (of $r$ pixels) of the secret image, each shadow image receives one of the generated pixels; the size of each shadow image is $1/r$ of the secret image. The following steps are the proposed secret image sharing method:

1. Truncate all gray values larger than 250 to 250 so that the gray values of the secret image are in the range 0–250.
2. Use a key to generate a permutation sequence to permute the pixels of the secret image.
3. Sequentially take $r$ not-shared-yet pixels of the permuted image to form a section.
4. Use the section in Step 3 and Eqs. (3) and (4) to generate $n$ pixels for the $n$ shadow images.
5. Repeat Steps 3 and 4 until all pixels of the permuted image are processed.

In the above, the permutation done in Step 2 is to increase the security. The integer (the so-called key) can be kept by the system owner or shared among the owners of shadows.

### 3.2. The reveal phase

The following steps are the reveal phase using any $r$ (of the $n$) shadow images:

1. Take the first non-used pixel from each of the $r$ shadow images.
2. Use these $r$ pixels (a subset of $\{q_j(1), \ldots, q_j(n)\}$) and the Lagrange's interpolation to solve for the coefficients $a_0-a_{r-1}$ in Eq. (3). The coefficients $a_0-a_{r-1}$ are then the corresponding $r$ pixel values of the permuted image.
3. Repeat Steps 1 and 2 until all pixels of the $r$ shadow images are processed.
4. Applying the inverse-permutation operation to the permuted image to get the secret image.

### 3.3. The lossless secret image sharing method

The method discussed in Section 3.1 is a lossy method because we must limit the gray value to the range 0–250 (see Step 1). To get a lossless image, we introduce below a special process to handle the gray values larger than 250. The processes will slightly increase the size of the shadow image. However, the increase is usually small because quite often there are only a few pixels whose gray values are 251–255. We slightly modify the method in Section 3.1 to get the lossless version. The following steps are the lossless secret image sharing method:

1. Use a key to generate a permutation sequence to permute the pixels of the secret image.
2. (Transform the permuted image $\tilde{D}$ to an array $E$.)
   Sequentially read in gray values of $\tilde{D}$ and then store in $E$ according to the rule below. For each read-in gray value $p_i$ of $\tilde{D}$:
   2.1. If $p_i < 250$, then store $p_i$ in $E$.
   2.2. If $p_i \geqslant 250$, then split $p_i$ into two values 250 and $(p_i - 250)$. Store these two values in $E$ (first 250, then $p_i - 250$).
3. Sequentially, take $r$ not-shared-yet elements of the array $E$ to form an $r$-pixel section.
4. Use the section in Step 3 and Eqs. (3) and (4) to generate $n$ pixels for the $n$ shadow images.
5. Repeat Steps 3 and 4 until all elements of the array $E$ are processed.

The following steps are the reveal phase for the lossless secret image sharing method:

1. Take the first non-used pixel from each of the $r$ shadow images.
2. Use these $r$ pixels (a subset of $\{q_j(1), \ldots, q_j(n)\}$) and the Lagrange's interpolation to solve for the coefficients $a_0 - a_{r-1}$ in Eq. (3). The coefficients $a_0 - a_{r-1}$ are then the values of the corresponding $r$ elements of the array $E$.
3. Repeat Steps 1–2 until all pixels of the $r$ shadow images are processed.
4. (Transform the array $E$ back to the permuted image $\tilde{D}$.)
   Sequentially read in the elements $\{e_i\}$ of the array $E$. For each read-in element $e_i$:
   4.1. If $e_i < 250$, then store $e_i$ in $\tilde{D}$. Now, delete $e_i$ from $E$ because the information contained in $e_i$ has been used.
   4.2. If $e_i = 250$, then read in $e_{i+1}$ immediately. Then store the single value $(250 + e_{i+1})$ in $\tilde{D}$. Now, delete from $E$ both $e_i$ and $e_{i+1}$ because the information contained in both $e_i$ and $e_{i+1}$ has been used.
5. Applying the inverse-permutation operation to the permuted image $\tilde{D}$ to get the secret image $D$.

## 4. The experimental result

The experimental result is shown in Fig. 1: (a) the secret image, and its data size is $512 \times 512$ bytes; (b) he permuted image of (a); (c)–(f) the four shadow images generated by using proposed method with $r = 2$ and $n = 4$. Each of these four has data size $512 \times 256$ bytes; (g) the image revealed by any two of the four shadow images and (g) is identical to (a). Because the gray values of all pixels of the original secret image (a) are all smaller than 250, the adjustment process (to split one gray value into two) in Step 2b is never used; the size of each shadow image is therefore just half of the secret image. In fact, many famous images, including "Lena", "Pepper" and "Baboon", have their all gray values less than 250. Therefore, we may expect the sizes of the shadow images of our method are often only $1/r$ of the secret image (here, $r = 2, 3, \ldots, n$).

## 5. Security analysis

In this section we will show that our method satisfies Feature 3 of Section 1: "Any $r - 1$ or less shadows cannot get sufficient information to reveal the secret image." For a $512 \times 512$ secret image, because each section is formed of $r$ pixels, there are $512 \times 512/r$ sections (see Step 3 of Section 3.1), in other words, there are $512 \times 512/r$ polynomials $\{q_j(x)\}_{1 \leqslant j \leqslant 512 \times 512/r}$. Without loss of generality, let us inspect how $q_1(x)$ can be revealed. From Eq. (3), to reveal the $r$ pixels (or the $r$ coefficients) $a_0 - a_{r-1}$ of the polynomial $q_1(x)$, we need $r$ equations. If we only have $r - 1$ shadow images, without loss of generality, say, we only have $q_1(1), q_1(2), \ldots, q_1(r-1)$, then we can construct only $r - 1$ equations

$$\begin{aligned}
(a_0 + a_1 + \cdots + a_{r-1}) \bmod 251 &= q_1(1),\\
(a_0 + 2a_1 + \cdots + 2^{r-1}a_{r-1}) \bmod 251 &= q_1(2),\\
&\vdots\\
(a_0 + (r-1)a_1 + \cdots + (r-1)^{r-1}a_{r-1}) \bmod 251 &= q_1(r-1).
\end{aligned}$$

To solve for $r$ unknowns using these $r - 1$ equations, there are 251 possible solution sets $\{(a_0, \cdots, a_{r-1})^{(k)}\}_{k=1}^{251}$; the first set corresponds to $a_0 = 0$, the second set corresponds to $a_0 = 1, \ldots$, and the 251th set corresponds to $a_0 = 250$. The possibility of guessing the right solution is only $1/251$. Now, there are $512 \times 512/r$ polynomials (for example, $512 \times 512/2 = 131{,}072$ polynomials if $r = 2$, as in Fig. 1). The possibility of obtaining the right image is only

$$\left(\frac{1}{251}\right)^{512 \times 512/r}.$$

Also note that our secret image has been permuted before doing any sharing process; therefore, there is no correlation between polynomials. In other words, the lack of information cannot be supplied from the image property such as the neighboring pixels are usually similar. Since $1/251^{(512 \times 512/r)}$ is a very small number, the hacker only has this small chance of the probability to obtain the permuted image (for example, Fig. 1(b)), not to mention the difficulty to obtain exactly the original image.

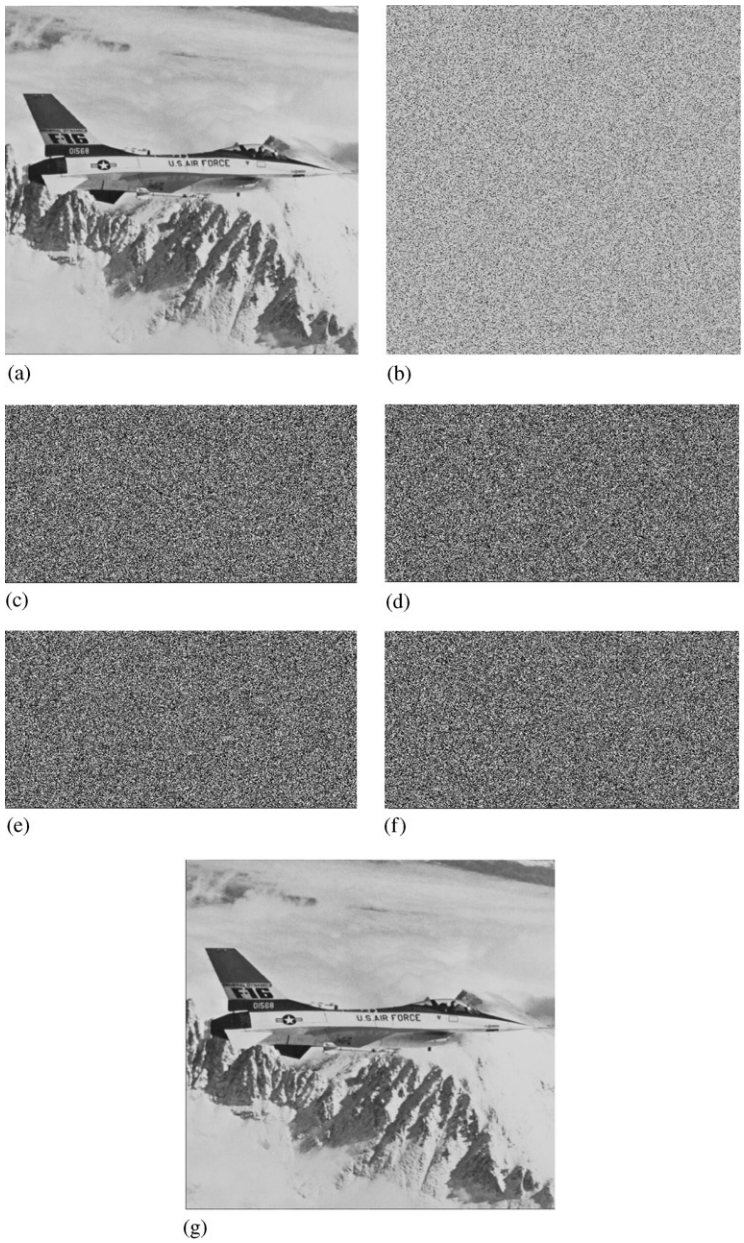


Fig. 1. Experimental results: (a) a $512 \times 512$ secret image; (b) the permuted image of (a); (c)–(f) the four shadow images whose size are all $512 \times 256$ bytes; (g) the image revealed from any two of these four images.

## 6. The benefits of the size reduction property of shadow images

One of the reasons to use secret sharing is to protect the secret from being lost or destroyed. We may apply the $(r, n)$ threshold scheme directly to the secret image, but the size of each shadow image would be the same as the secret image. On the other hand, our method can reduce the size of each shadow image to $1/r$ of the secret image's ($r = 2, 3, \ldots, n$). The small size of each shadow image is a good property in practice. Besides the saving of storage space or transmission time, some other benefits also exist. For example, we can easily use data hiding technique to hide each shadow image in some other images called host images (for example, hiding Fig. 1(c) in the image "Lena") so that the thief will not

notice the existence of the shadow image Fig. 1(c) when he saw a lot of images in our database among them "Lena" is one of the images. One of the reviewers suggested that hiding the $n$ shadows in $n$ different host images instead of one single host image to avoid the attention of hackers (the hackers might use statistics technique to detect possible images carrying information, see Ref. [9]). Figs. 2(a)–(d) show, respectively, the resulting images of embedding Fig. 1(c)–(f) in four different images; none of the resulting images look abnormal. Note that many hiding methods (e.g. [7,8]) require that the image to be hidden in a host image be at least 1/2 (or even 1/4) smaller than the size of the host image. The small size of our shadow images is therefore convenient. Also note that image hiding is desired even if Shamir's method is used, because the shadow images produced by that method will also have non-natural appearance (looks like Fig. 1(b) instead of Fig. 1(a)), so, without the cover of image hiding, the shadow images tend to get more attention from a thief once the thief break in an image database system in which most pictures have natural look (such as "Lena").

## 7. Conclusions

We proposed a method such that secret image can be shared by several shadow images. The size of each shadow image is $1/r$ of the secret image's in our method, and this small size property gives our method certain benefits including easier process for storage, transmission, and hiding. Two versions are provided to overcome the problem that some pixels of the secret image might have gray values not less than 250.

As a final remark, some dot-com companies recently have provided free "network hard disk" with limited storage space for general users. Therefore, a user may have several network hard disks for storage. These network hard disks could be treated as the distributed storage, and each network hard disk just store one of the (smaller size) shadows. The proposed method is especially valuable here when the storage space of a single dot-com account is not enough to store the whole image.

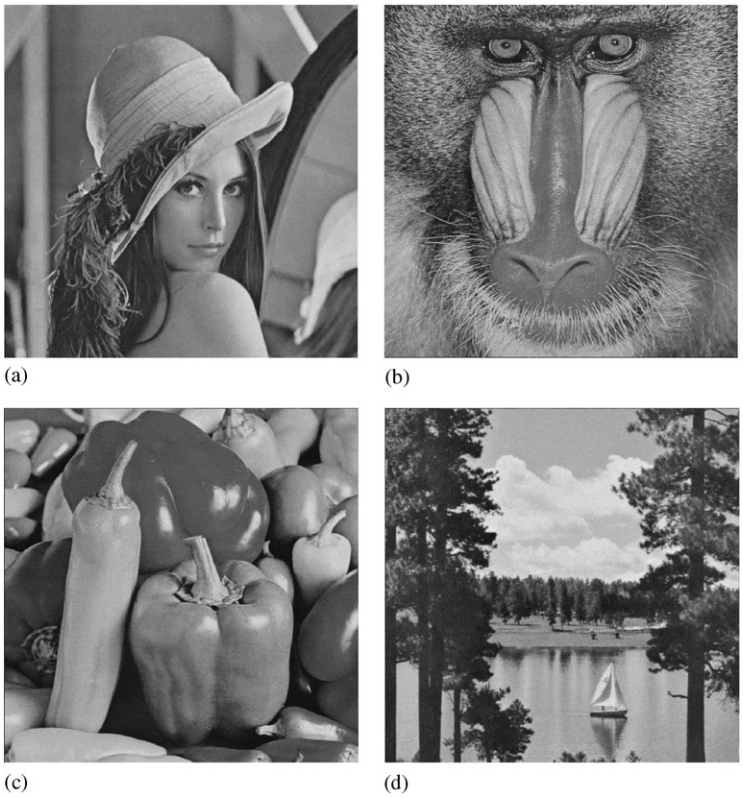


Fig. 2. Hiding shadows in some other existing images: (a)–(d) the images obtained after embedding the shadows Fig. 1(c)–(f), respectively, in the images: Lena, Monkey, Peppers, and Scene.

## Acknowledgements

The authors would like to thank the two reviewers for their valuable comments.

## References

[1] Blakley GR. Safeguarding cryptographic keys. Proceedings AFIPS 1979 National Computer Conference, vol. 48, New York, USA, 4–7 June 1979. p. 313–7.
[2] Shamir A. How to share a secret. Communication of the ACM 1979;22(11):612–3.
[3] Ahlswede R, Csiszàr I. Common randomness in information theory and cryptography—part I: secret sharing. IEEE Transactions on Information Theory 1993;39(4):1121–32.
[4] Stinson DR. Decomposition constructions for secret-sharing schemes. IEEE Transactions on Information Theory 1994;40(1):118–24.
[5] Beimel A, Chor B. Universally ideal secret-sharing schemes. IEEE Transactions on Information Theory 1994;40(3): 786–94.
[6] Beimel A, Chor B. Secret sharing with public reconstruction. IEEE Transactions on Information Theory 1998;44(5):1887–96.
[7] Wu DC, Tsai WH. Spatial-domain image hiding using image differencing. IEE Proceedings: Vision, Image & Signal Processing 2000;147(1):29–37.
[8] Chae JJ, Mukherjee D, Manjunath BS. A robust data hiding technique using multidimensional lattices. IEEE International Forum on Research and Technology Advances in Digital Libraries, Santa Barbara, CA, USA, 22–24 April 1998. p. 319–26.
[9] Avcibas I, Memon N, Sankur B. Steganalysis of watermarking techniques using image quality metrics. Proceedings of SPIE, vol. 4314, San Jose, CA, USA, 22–25 January 2001. p. 523–31.