Optical Engineering

OpticalEngineering.SPIEDigitalLibrary.org

All-in-one visual and computer decoding of multiple secrets: translated-flip VC with polynomial-style sharing

Chia-Hua Wu
Suiang-Shyan Lee
Ja-Chen Lin

SPIE.

Chia-Hua Wu, Suiang-Shyan Lee, Ja-Chen Lin, "All-in-one visual and computer decoding of multiple secrets: translated-flip VC with polynomial-style sharing," *Opt. Eng.* **56**(6), 063106 (2017), doi: 10.1117/1.OE.56.6.063106.

Optical Engineering 56(6), 063106 (June 2017)

All-in-one visual and computer decoding of multiple secrets: translated-flip VC with polynomial-style sharing

Chia-Hua Wu, Suiang-Shyan Lee, and Ja-Chen Lin*
National Chiao Tung University, Department of Computer Science, Hsinchu, Taiwan

Abstract. This all-in-one hiding method creates two transparencies that have several decoding options: visual decoding with or without translation flipping and computer decoding. In visual decoding, two less-important (or fake) binary secret images S_1 and S_2 can be revealed. S_1 is viewed by the direct stacking of two transparencies. S_2 is viewed by flipping one transparency and translating the other to a specified coordinate before stacking. Finally, important/true secret files can be decrypted by a computer using the information extracted from transparencies. The encoding process to hide this information includes the translated-flip visual cryptography, block types, the ways to use polynomial-style sharing, and linear congruential generator. If a thief obtained both transparencies, which are stored in distinct places, he still needs to find the values of keys used in computer decoding to break through after viewing S_1 and/or S_2 by stacking. However, the thief might just try every other kind of stacking and finally quit finding more secrets; for computer decoding is totally different from stacking decoding. Unlike traditional image hiding that uses images as host media, our method hides fine gray-level images in binary transparencies. Thus, our host media are transparencies. Comparisons and analysis are provided. [DOI: 10.1117/1.OE.56.6.063106]

Keywords: multiple decoding options; multiple secrets; visual cryptography; polynomial-style sharing.

Paper 170071 received Jan. 11, 2017; accepted for publication Jun. 1, 2017; published online Jun. 21, 2017.

1 Introduction

Visual cryptography (VC)[1] is a well-known technique to hide secret information. The simplest VC scheme splits a binary secret image into two transparencies. Each transparency looks like random noise. However, the secret image can be revealed by stacking the two transparencies. Secret disclosure is through human vision rather than by a computer. Consequently, secret decoding is very fast in VC. In VC, each transparency is often larger than the input secret image. This phenomenon is called pixel expansion.

Recently, many studies[2–8] focused on hiding multiple secrets in two transparencies. Translation operation was employed in some studies[2–5] to decode multiple secrets. One of the two transparencies is shifted to a specific position before stacking with the other transparency. In addition to hiding a secret image, Fang and Lin[2] also hid a confidential text file describing the secret image, and a watermark image is utilized in Ref. 5 to verify the authenticity. Lin et al.[6] hid two images using flip VC. One of the secret images can be revealed by stacking two transparencies. The other secret image is revealed by flipping one of the two transparencies and then stacking with the other transparency. Shyu et al.[7] hid more than two secrets in two circular transparencies. Later, Shyu and Chen[8] elegantly proposed VC schemes using flipping and turning to hide multiple secrets. Both the VC scheme gorgeously proposed by Katta[9] and the VC scheme cleverly designed by Kumar and Sharma[10] used recursive translations to hide many secret images recursively. Notably, flipping and translation were not used simultaneously in any of the earlier works.[1–10]

Although these reported VC works can hide multiple secrets in a few transparencies, once an intruder obtains all transparencies stored in distinct companies or distinct cities, he can try every manner to stack transparencies and may be successful in unveiling all secrets. Therefore, we propose a method in this study wherein the most important secrets are not unveiled by stacking transparencies, while some less important secrets (including some fake secrets made on purpose to cheat or confuse the intruder) are unveiled by stacking transparencies using several stacking methods. In our design, low-level or fake secret images $\{S_1, S_2\}$ can be decoded by stacking transparencies as follows: (1) the fake or least important image S_1 is revealed by the direct stacking of two transparencies and (2) the fake or moderately important image S_2 is revealed by translation-flip stacking. Meanwhile, high level secrets can be revealed only through a computer using the information hidden in the two transparencies. Since unveiling the most important secret requires a completely distinct method and tool, a strategy using our design to cheat or confuse an intruder shares two fake secret images in two transparencies to mislead the intruder into thinking that he has disclosed all secrets. The detail is explained below. The intruder might think that he has found all secrets when he finds the second fake secret because unveiling the second fake secret entails many attempts to obtain a suitable translation-flip stacking. Even if the thief does not stop at the second image, he will probably attempt other ways of stacking the transparencies to find the third secret. After the useless but time-consuming tries again and again, he will probably give up. Occasionally, if he does not want to give up, and if he knows to switch to computer decoding, then he would need the parameter value of the key.

Below we discuss the issue of host media. People often use image hiding techniques to hide fine secrets in images. Here we use two transparencies rather than images to hide "high-quality" secret image S_3. Fine 256-level gray-value

*Address all correspondence to: Ja-Chen Lin, E-mail: jclin@cs.nctu.edu.tw

0091-3286/2017/$25.00 © 2017 SPIE

images or even 24-bit color images can be hidden in our two transparencies. We review polynomial-style sharing (PSS) here because our computer decoding is based on it. Thien and Lin[11] proposed a PSS scheme. Their outputs were several files called shares rather than transparencies. Shares should be stored in distinct places. Although polynomial decryption requires additional computation, share size of PSS is more economical than the transparency created by VC schemes. Therefore, the PSS scheme has been applied to other extended applications, such as the essential scheme, weighted scheme, and progressive scheme. The readers who are interested in this topic can consult the informative introduction in Ref. 12. A big difference between VC and PSS is that VC can obtain rough decoding instantly without a computer, whereas PSS can obtain a fine image using a computer. To date, both VC and PSS schemes are important branches of information sharing, and they can be applied in many fields for protection, such as in medicine,[13,14] computer graphics,[15,16] and computer networks.[17]

Our system has a VC version (Sec. 3) and a VC + PSS version (Sec. 4). When an intruder steals both transparencies from two distinct places, the version in Sec. 4 provides strong confusion protection of the true secret. Here, confusion protection means that the intruder might think that he has viewed all secrets and no further secret exists. More discussion about the confusion protection can be found in Sec. 6.6. As a remark, the flip VC in Ref. 6, which also used the flipping operation, exhibited neither a confusion property nor a computer decoding option. Moreover, its flipping is without the combination of any translation; and hence, gives no protection when a thief steals both transparencies.

The rest of this study is organized as follows. Section 2 reviews two VC schemes: basic and flip VC. We propose a translated-flip VC scheme with no pixel expansion in Sec. 3. The scheme can hide two binary secret images as follows. Secret 1 can be decoded in a traditional way by direct stacking, whereas secret 2 is decoded through special stacking: one transparency is translated to a user-specified pixel and the other one is flipped before stacking. Section 4 proposes an all-in-one double-decoding-options (visual decoding versus computer decoding) method. The two transparencies in Sec. 4 can hide not only two binary secret images but also some other secret files. Section 5 lists the experimental results. Section 6 contains the discussion and comparison. Section 7 provides the conclusions.

2 Related Works

2.1 *Conventional Visual Cryptography*

Naor and Shamir[1] presented the concept of the VC scheme. They introduced how to generate two transparencies for a given binary secret image. The transparencies are stacked together to view the secret image. However, each transparency shows nothing but noise. Their solution is based on two matrices C_W and C_B. Equations (1) and (2) list an example of the two basis matrices for the VC scheme

$$C_W = \left\{ \text{permute columns of} \begin{bmatrix} 1 & 0 \\ 1 & 0 \end{bmatrix} \right\} = \left\{ \begin{bmatrix} 1 & 0 \\ 1 & 0 \end{bmatrix}, \begin{bmatrix} 0 & 1 \\ 0 & 1 \end{bmatrix} \right\}, \quad (1)$$

$$C_B = \left\{ \text{permute columns of} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \right\} = \left\{ \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \right\}. \quad (2)$$

Individuals can encrypt a white (or black) pixel of the secret image by randomly selecting one of the two matrices. Then, each row i of the chosen matrix is assigned to the i'th transparency. Here, 0 represents a white pixel and 1 represents a black pixel. Clearly, a pixel of the secret image yields two elements (i.e., one row) in a transparency. Thus, the created transparency has a pixel expansion rate of 1:2. The VC design is equivalent to the design of the two basis matrices C_W and C_B, which are for encoding of the white and black pixels, respectively.

2.2 *Flip VC*

Lin et al.[6] introduced flip visual cryptography (FVC). Two input $N \times M$ binary secret images $S_1 = \{s_1(i,j) | 0 \le i < N, 0 \le j < M\}$ and $S_2 = \{s_2(i,j) | 0 \le i < N, 0 \le j < M\}$ are processed to obtain two output $M \times N$ transparencies $T_1 = \{t_1(i,j) | 0 \le i < N, 0 \le j < M\}$ and $T_2 = \{t_2(i,j) | 0 \le i < N, 0 \le j < M\}$. Image S_1 can be decoded by stacking T_1 and T_2 together. S_2 can be decoded by stacking T_2 with T_1^{Flip}, where T_1^{Flip} denotes the flipped T_1.

In Ref. 6, they defined 16 basis matrices. These 16 matrices were then used to paint the two transparencies. The produced transparencies exhibited no pixel expansion. Their basis matrices were based on the probabilistic method. The white and black pixels in the transparencies possessed similar probability distributions. Thus, no information leak was observed. When stacking transparencies, the white pixels of the secret images have a $1/6$ chance of being unveiled as white pixels. Lin et al.[6] proved that their contrast value is $1/6$. They also proved that this contrast value is conditionally optimal if (a–b) are both true: (a) without pixel expansion and (b) each transparency does not leak any information about the pixel values or the relation among secret images.

3 Preprocessing: Our Design of a Translated FVC

In this section, we design a translated-flip VC, abbreviated as "translated FVC," that generates two transparencies. These two created transparencies will be used later as the input resource in Sec. 4. The basic concept of the design is as follows: two secret images S_1 and S_2 are provided, and S_1 is larger than S_2. We can create two transparencies T_1 and T_2 using inputs S_1 and S_2. Both are as large as S_1 (recall that S_1 is larger than S_2). Secret image S_1 can be decoded by stacking two transparencies directly. The other secret image (S_2) can be decoded by three steps: (1) flipping one of the two transparencies, (2) translating the other transparency, and (3) stacking the two transparencies. Therefore, this VC scheme is called the "translated FVC," where "F" indicates that the VC involves a flipping operation.

For the two input secret binary images S_1 and S_2, the large one is assumed an $N \times M$ image $S_1 = \{s_1(i,j) | 0 \le i < N, 0 \le j < M\}$. The small one is an $N' - \text{by} - M'$ image $S_2 = \{s_2(i,j) | 0 \le i < N', 0 \le j < M'\}$, where $M' \le M$ and $N' \le N/2$. We create two $N \times M$ output binary transparencies as follows:

$$T_1 = \{t_1(i,j)|0 \le i < N, 0 \le j < M\} \quad \text{and}$$
$$T_2 = \{t_2(i,j)|0 \le i < N, 0 \le j < M\}. \tag{3}$$

The horizontal and vertical differences between the sizes of the two secret images are

$$h = M - M' \tag{4}$$

and

$$v = N - N'. \tag{5}$$

Let $S_1' = \{s_1'(i,j)|0 \le i < N, 0 \le j < M\}$ be the stacking result that represents the binary secret image S_1. Let $S_2' = \{s_2'(i,j)|0 \le i < N', 0 \le j < M'\}$ be the stacking result that represents S_2. We require the following because of the double demands of stacking:

1. $s_1(i,j)$ is decoded by stacking $t_1(i,j)$ and $t_2(i,j)$, where $0 \le i < N$, $0 \le j < M$;
2. $s_2(i,j)$ is decoded by stacking $t_1(i, M-1-j)$ and $t_2(i+v, j+h)$, where $0 \le i < N'$, $0 \le j < M'$.

We start our design using the s_2 information because the decoding of the s_2 secret is slightly harder than that of the s_1 secret. s_2 decoding is accomplished through T_1 flipping and the (h,v) unit translation of T_2 before stacking. Without the loss of generality, we may assume that

$$s_2'(i,j) = t_1(i, M-1-j) \vee t_2(i+v, j+h), \quad \text{where } 0 \le i < N', \quad 0 \le j < M'. \tag{6}$$

In this paper, the OR logic operator "$\vee$" is utilized to realize the stacking operation. In other words, $\{0 \vee 0 = 0, 0 \vee 1 = 1, 1 \vee 0 = 1, 1 \vee 1 = 1\}$. $M-1-j$ is a mapping result of flipping an index value j to a value $M-1-j$ by the flipping mapping $\{0 \to (M-1), 1 \to (M-1)-1, 2 \to (M-1)-2, \ldots, j \to (M-1)-j, \ldots, (M-1) \to 0\}$. Thereafter, in the design of the two transparencies, if we already used both Eq. (6) and the small secret image S_2 to paint both $N'-$ by $-M'$ regions

$$\{t_1(i, M-1-j): 0 \le i < N', 0 \le j < M'\} \text{ of transparency } T_1, \tag{7}$$

and the $N'-$ by $-M'$ region

$$\{t_2(i+v, j+h): 0 \le i < N', 0 \le j < M'\} \text{ of transparency } T_2. \tag{8}$$

Then, as the pixel values in Eq. (7) are determined, we can use secret image S_1 and the equation

$$s_1'(i, M-1-j) = t_1(i, M-1-j) \vee t_2(i, M-1-j), \quad \text{where } 0 \le i < N', \quad 0 \le j < M', \tag{9}$$

to determine the other $N'-$ by $-M'$ binary pixel values of transparency T_2, that is, the $\{t_2(i, M-1-j): 0 \le i < N', 0 \le j < M'\}$ of transparency T_2. Similarly, because the pixel values in Eq. (8) are already determined, we can use

$$s_1'(i+v, j+h) = t_1(i+v, j+h) \vee t_2(i+v, j+h), \quad \text{where } 0 \le i < N', \quad 0 \le j < M', \tag{10}$$

to paint the other $N'-$ by $-M'$ binary pixel values of transparency T_1, that is, the $\{t_1(i+v, j+h): 0 \le i < N', 0 \le j < M'\}$ of transparency T_1. To date, two $N'-$ by $-M'$ regions of transparency T_1 are already painted, which are $\{(i, M-1-j): 0 \le i < N', 0 \le j < M'\}$ and $\{(i+v, j+h): 0 \le i < N', 0 \le j < M'\}$. Similarly, the area of transparency T_2 is also painted. The remaining $(NM - 2N'M')$ pixels of T_1 [and the same $(NM - 2N'M')$ pixels of T_2] can be easily painted using the condition that stacking T_1 and T_2 will obtain secret S_1.

In accordance with the above analysis, we create eight basis matrices in Table 1. The subscripts of these eight basis matrices $\{C_{WWW}, C_{WWB}, C_{WBW}, C_{WBB}, C_{BWW}, C_{BWB}, C_{BBW}, C_{BBB}\}$ correspond to the eight possible combinations of the triple pixels $[s_1(i+v, j+h), s_1(i, M-1-j)$, and $s_2(i,j)]$ for $0 \le i < N'$ and $0 \le j < M'$. To use this table, for each (i,j) in the range $\{0 \le i < N', 0 \le j < M'\}$, we use the values of the triple pixels to obtain the corresponding matrix selected from the eight matrices. Then, we randomly select one of the six columns of the matrix and extract the four elements of the column. Thereafter, we use these four elements to paint the transparency pixels $t_1(i+v, j+h)$, $t_1(i, M-1-j)$, $t_2(i+v, j+h)$, and $t_2(i, M-1-j)$ for $0 \le i < N'$, $0 \le j < M'$, respectively. Therefore, we stack transparencies T_1 and T_2 to obtain secret S_1. In terms of the corresponding matrix in Table 1, we stack an element of row 1 with the same column element of row 3 to obtain $t_1(i+v, j+h) \vee t_2(i+v, j+h)$ or stack an element of row 2 with the same column element of row 4 to obtain $t_1(i, M-1-j) \vee t_2(i, M-1-j)$. Meanwhile, the stacking of rows 2 and 3 is related to translated-flipping stacking to obtain the secret image S_2. This is because our design is in accordance with $s_2'(i,j) = t_1(i, M-1-j) \vee t_2(i+v, j+h)$ as stated in Eq. (6).

To date, we have determined how to paint regions $\{(i, M-1-j): 0 \le i < N', 0 \le j < M'\}$ and $\{(i+v, j+h): 0 \le i < N', 0 \le j < M'\}$ for both T_1 and T_2. We can still refer to Table 1 to paint a pixel-pair $\{t_1(i,j)$ and $t_2(i,j)\}$ for a pixel (i,j) in the remaining area to achieve this result because the remaining area of T_1 and T_2 involves only secret S_1. The one-bit value of $s_1(i,j)$ should be used to determine the basis matrix. If $s_1(i,j)$ is "W", i.e., 0, then the left four matrices of Table 1 should be used because the first letter of their subscript is "W." Otherwise, the right four matrices should be used. Then, one matrix from these four matrices should be randomly selected. Subsequently, one column from the six columns of the matrix should be randomly selected. Then, we take only two elements, that is, using the element in row 1 of the column to paint $t_1(i,j)$ rather than taking all four elements. We also use the element in row 3 of the same column to paint $t_2(i,j)$. Table 1 ensures that if $s_1(i,j)$ is "B," i.e., 1, stacking rows 1 and 3 will always obtain 1. If $s_1(i,j)$ is "W," i.e., 0, stacking rows 1 and 3 will obtain 1 (probability is $5/6$) or obtain 0 (probability is $1/6$).

4 Translated FVC with PSS (T-FVC-PSS)

4.1 *Overview*

This section designs an all-in-one two-decoding-options method, that is, the "translated FVC with PSS." The

Table 1 Translated FVC system in Sec. 3. The $2^3 = 8$ basis matrices are determined by triple values $[s_1(i+v, j+h), s_1(i, M-1-j), s_2(i,j)]$ for $0 \le i < N'$, $0 \le j < M'$.

Reading of the input secret pixels $s_1(i+v, j+h)$, $s_1(i, M-1-j)$, $s_2(i,j)$	Basis matrix corresponding to the input quadruple secret pixels	Reading of the input secret pixels $s_1(i+v, j+h)$, $s_1(i, M-1-j)$, $s_2(i,j)$	Basis matrix corresponding to the input quadruple secret pixels
W, W, W,	$C_{WWW} = \begin{bmatrix} 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 & 1 & 0 \end{bmatrix}$	B, W, W	$C_{BWW} = \begin{bmatrix} 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 1 & 1 \end{bmatrix}$
W, W, B	$C_{WWB} = \begin{bmatrix} 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 & 1 & 0 \end{bmatrix}$	B, W, B	$C_{BWB} = \begin{bmatrix} 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 0 \end{bmatrix}$
W, B, W	$C_{WBW} = \begin{bmatrix} 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 & 1 & 0 \end{bmatrix}$	B, B, W	$C_{BBW} = \begin{bmatrix} 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 1 \end{bmatrix}$
W, B, B	$C_{WBB} = \begin{bmatrix} 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 & 0 & 1 \end{bmatrix}$	B, B, B	$C_{BBB} = \begin{bmatrix} 0 & 1 & 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 & 1 \end{bmatrix}$

encoding phase uses the two (already encoded) transparencies generated in Sec. 3 as input, and then creates two new enlarged transparencies by combining other f secret files together with the two transparencies generated in Sec. 3. Decoding is the inverse procedure of encoding. Users have two decoding options using the new transparencies created in Sec. 4. Decoding option 1 is using human vision directly. That is, binary secret image S_1 can be revealed by stacking the two new transparencies together, whereas the other binary secret image S_2 can be revealed by flipping-and-translation before stacking. Decoding option 2 requires a computer. This method can recover in detail other f secret files, such as some fine gray-value images or confidential military plan. Figures 1 and 2 show the flowcharts of the encoding and decoding phases, respectively.

4.2 *Encoding Phase*

4.2.1 *Transparency expansion*

We expand the two transparencies $\{T_1, T_2\}$ created in Sec. 3 to hide additional data (for example, confidential files and secret gray-value images). Each pixel in a transparency $T \in \{T_1, T_2\}$ is scaled up to a 2×2 block $[\, a_0 \quad a_1 \quad a_2 \quad a_3 \,]$. In other words, if the original size of transparency T is $N \times M$, the size of the corresponding output transparency T' would be $2N \times 2M$. Each a_i of each 2×2 block $[\, a_0 \quad a_1 \quad a_2 \quad a_3 \,]$

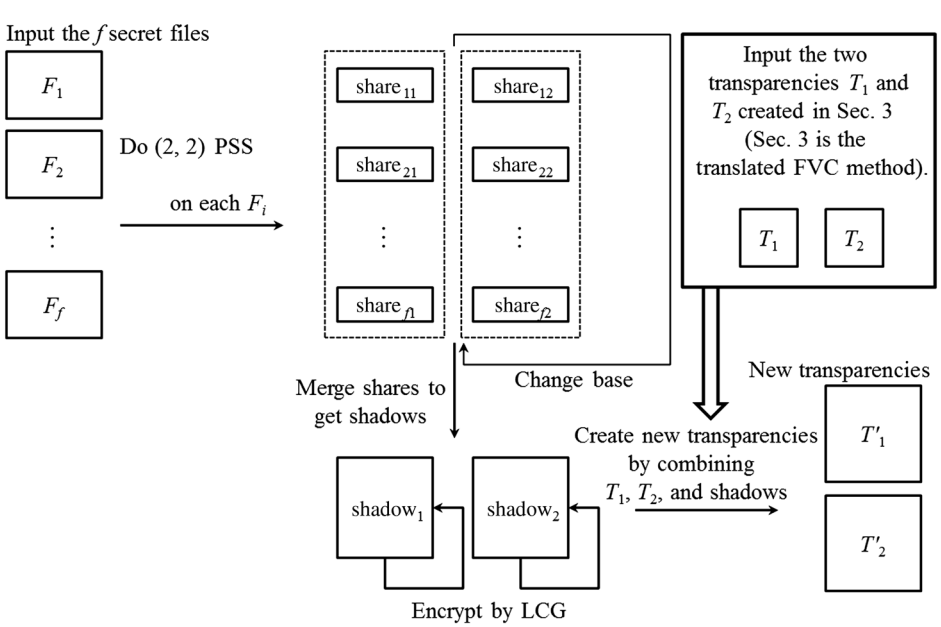

Fig. 1 Flowchart of encoding phase of Sec. 4 (translated FVC with PSS).

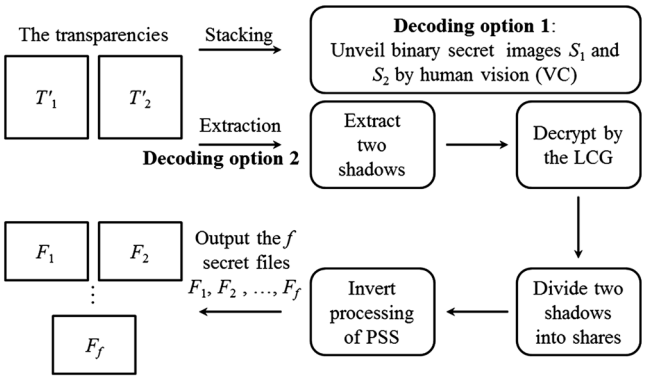

Fig. 2 Flowchart of decoding phase of Sec. 4 (translated FVC with PSS).

is either 1 (black) or 0 (white). Thus, $2^4 = 16$ types $\{[0\ 0\ 0\ 0], [0\ 0\ 0\ 1], [0\ 0\ 1\ 0], \ldots, [1\ 1\ 1\ 1]\}$ of blocks exist. Some might try to use these 16 types of blocks to hide a secret number. For example, let {[0 0 0 0], [0 0 0 1], [0 0 1 0], and ..., [1 1 1 1]} represent {type 0, type 1, type 2, ..., type 15}, respectively. Then, these can be used to hide any base-16 secret number. Unfortunately, using all 16 types will destroy the visual decoding of the secret image. Option 1 of our double-decoding goal, which is stacking decoding, should not be forgotten. In general, the black area should look darker than the white area. The 16-type expansion often destroys this visual requirement after hiding a secret file.

For each a_i of the four elements of a 2×2 expansion $[a_0\ \ a_1\ \ a_2\ \ a_3]$ of a pixel, $a_i = 1$ will paint black (impervious) in the transparency, while $a_i = 0$ will paint white (transparent) in transparency. In our design, to produce a black block appearance that is darker than the white block, we require

$$a_0 + a_1 + a_2 + a_3 > 2 \quad \text{for the expansion of a black pixel,} \tag{11}$$

$$a_0 + a_1 + a_2 + a_3 < 2 \quad \text{for the expansion of a white pixel.} \tag{12}$$

If the 2×2 block is in accordance with Eq. (11), then $[a_0\ \ a_1\ \ a_2\ \ a_3]$ is a black block and contains one zero or no zero. Thus, the five types ($5 = 1 + 4 = C_0^4 + C_1^4$) of black blocks are

$$\{B_0, B_1, B_2, B_3, B_4\} = \{[1\ 1\ 1\ 1], [0\ 1\ 1\ 1], [1\ 0\ 1\ 1], [1\ 1\ 0\ 1], [1\ 1\ 1\ 0]\}. \tag{13}$$

Likewise, if the 2×2 block is in accordance with Eq. (12), this block is then a white block and possesses at least three zeros. The five types ($5 = 1 + 4 = C_4^4 + C_3^4$) of white blocks are

$$\{W_0, W_1, W_2, W_3, W_4\} = \{[0\ 0\ 0\ 0], [1\ 0\ 0\ 0], [0\ 1\ 0\ 0], [0\ 0\ 1\ 0], [0\ 0\ 0\ 1]\}. \tag{14}$$

Figure 3 lists the five types of black blocks $\{B_0, B_1, B_2, B_3, B_4\}$ to expand a black pixel and the five types of white blocks $\{W_0, W_1, W_2, W_3, W_4\}$ to expand a white pixel. We can hide a value in the range of 0 to 4 when we expand a black pixel (or a white pixel) using this table. If the secret file is a gray-value image, we can convert the gray-value image to a numerical file where each digit is 0, 1, 2, 3, or 4. If this process is troublesome, the other method is to deal with each gray value directly in a pixel-by-pixel hiding manner. In other words, let x be the number of 2×2 blocks that we require when we want to hide a gray value in the range of 0 to 255. Then,

$$5^x \geq 256, \tag{15}$$

is necessary. The minimum integer solution for x is 4. As a result, we may say that when we expand a black pixel (or a white pixel, respectively) to hide a secret value, we use one of the five possible black blocks (or one of the five possible white blocks, respectively) to hide a secret value in the range of 0 to 4. Likewise, the expansion generates "four blocks" of 2×2 each when we expand "four pixels" (for example, BWWB) of a transparency to hide a secret gray value in the range of 0 to 255. The four transparency-pixels BWWB will be expanded as four 2×2 blocks B_0-W_2-W_1-B_0 (Fig. 3) if the eight bits of a given to-be-hidden secret gray value (in the range of 0 to 255) is 00-10-01-00.

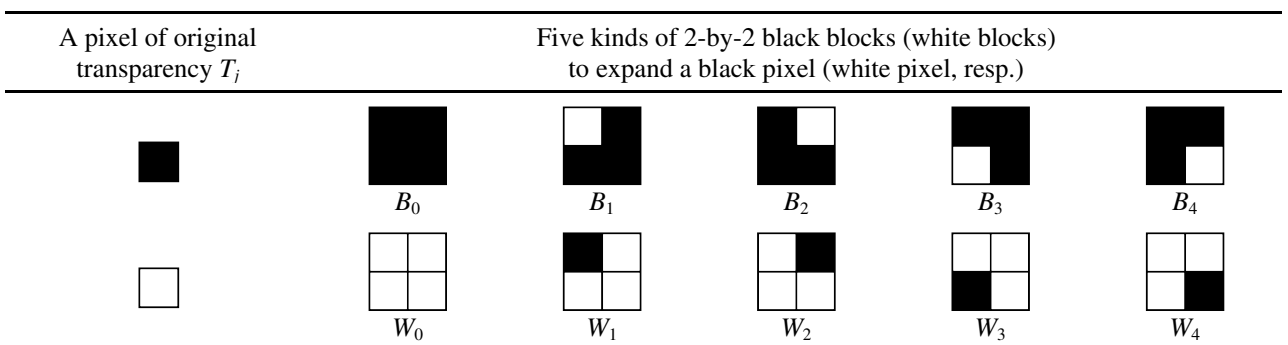

Fig. 3 Five kinds of black (white) blocks to expand a black (white, respectively) pixel of transparency T_j.

4.2.2 *Using PSS to create shadows that represent the (extra) secret file*

Each shadow contains parts of data from the f input secret files $\{F_1, F_2, \ldots, F_f\}$. Attackers cannot reveal any secret file if he cannot obtain enough shadows. Here, we use two well-known techniques to create our shadows. One of them is the PSS. The other is the linear congruential generator (LCG) algorithm.

In the literature, a (k, n) PSS[11,16,18] can share a secret file among n shares. The secret file can be revealed using k of the n shares, whereas less than k shares reveal nothing. Notably, each share is a string of digits in the range $\{0, 1, \ldots, 255\}$ because the Galois field of $2^8 = 256$ was used in our sharing process, that is, each share is a byte string because eight bits is a byte and $2^8 = 256$. Then, we use $(k = 2, n = 2)$ PSS to share each secret file F_i among $n = 2$ shares. Using $(k, n) = (4, 4)$, (6, 6), (8, 8), and so on is also acceptable. As $i = 1, 2, \ldots, f$, we have $2f$ shares: $\{i1$ and $i2 : i = 1, \ldots, f\}$. Then, we bind these $2f$ shares to obtain two shadows. Here, shadow$_1$ is the union of all f shares $\{i1 : i = 1, \ldots, f\}$. Shadow$_2$ is the union of all f shares $\{i2 : i = 1, \ldots, f\}$. We want to hide shadows in transparencies using five block types. Thus, we transform each shadow$_i$ from a byte string, that is, a digit string of base 256, to a new digit string shadow$_i'$ of base 5.

We may encrypt each shadow using the LCG algorithm before hiding the shadows in the transparencies to increase the security level. The readers can replace the LCG algorithm by any data-encryption algorithm they prefer. The LCG algorithm generates a sequence of pseudorandom values. We utilize it to return a shuffled copy of our shadow. The generator is defined by the recurrence relation

$$X_{i+1} = (A \times X_i + B) \bmod P, \tag{16}$$

where X is the sequence of random values. A, B, and P are the integer constants used in the generator. Let P be the size of shadow$_j$, that is, $|\text{shadow}_j|$. We generate a sequence X, which has $|\text{shadow}_j|$ members. Then, we shuffle the value of the i'th position to the $(X_i + 1)$'th position for $i = 1, 2, \ldots, |\text{shadow}_j|$.

4.2.3 *Combining the transparencies and shadows*

We combine input transparencies T_1 and T_2 with shadow$_1$ and shadow$_2$ to create the new transparencies after gathering the shadows, namely, the expanded transparencies T_1' and T_2'. Based on Eqs. (13) and (14), we design the following two 4×5 basis matrices:

$$C_{\mathrm{B}} = \begin{bmatrix} 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 0 \end{bmatrix} \quad \text{for black expansion;}$$

$$C_{\mathrm{W}} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \quad \text{for white expansion.} \tag{17}$$

Matrix C_{B} is for the expansion of black pixels. Matrix C_{W} is for the expansion of white pixels. Each matrix possesses five columns. Column i is related to block type i. For example, to create the first block of T_j', where $j \in \{1, 2\}$, the first pixel of T_j determines whether the matrix is C_{B} or C_{W}. Then, the first digit (which is a value of base 5) of shadow$_j$ determines which of the five columns of the matrix in Eq. (17) should be used. Each column contains four elements. These four elements are used to paint the first 2×2 block of T_j'. Every digit of shadow$_j$ is in the range [0, 4], which is of base 5. As we proceed sequentially, we can obtain the whole T_j'. Figure 4 shows an example of combining a transparency created in Sec. 3 with a shadow created in Sec. 4.2.2 to obtain a new (expanded) transparency.

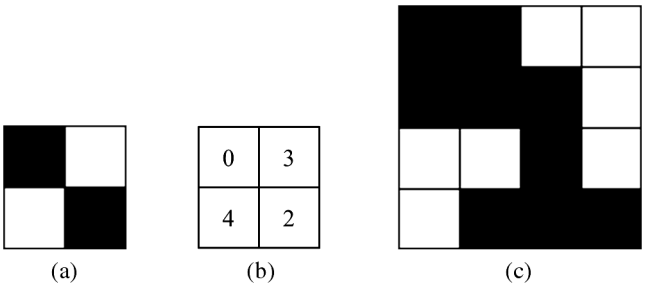

Fig. 4 Example of combining four pixels of an input transparency T_j with a byte of shadow$_j$ to obtain blocks of the expanded transparency T_j'. (a) is assumed as four pixel area of transparency T_j created in Sec. 3 for two black-and-white secret images; (b) is assumed as a byte (already transformed from $0 \times 5^3 + 3 \times 5^2 + 4 \times 5 + 2 = 97$ of base 256 to 0-3-4-2 of base 5) of shadow$_j$ created in Sec. 4.2.2 for extra secret file F; and (c) is the created four blocks (2×2 each) of the expanded output transparency T_j'.

4.3 *Decoding Phase*

VC decoding using T_1' and T_2' is similar to VC decoding using T_1 and T_2. In other words, secret S_1 should be revealed by stacking. S_2 should be revealed by translation flipping before stacking. Below we explain how to unveil extra secret files F_i for $i = 1, 2, \ldots, f$.

First, T_1' and T_2' must be split into 2×2 blocks. Then, each 2×2 block corresponds to one column of the basis matrices in Eq. (17). Columns 0 to 4 mean digits 0 to 4 from left to right of each basis matrix because the basis matrices are 4×5 and one column corresponds to one digit. We obtain two shadows, shadow$_1$ and shadow$_2$, after this mapping.

Second, we have to decrypt the two shadows because they were shuffled using the LCG algorithm. Three parameters A, B, and X_0 are the keys to generate sequence X, which was used in the encryption of Eq. (16). We shuffle the $(X_i + 1)$'th digit of the digit string to the i'th position for $i = 1, 2, \ldots, |\text{shadow}_j|$. Then, encrypted shadows become decrypted shadows.

Subsequently, we transform the shadows from base 5 to base 256. Then, each shadow$_j$ must be split to obtain share$_{ij}$ for each $i = 1, 2, \ldots, f$ and $j = 1, 2$. Finally, the secret file F_i from share$_{i1}$ and share$_{i2}$ must be recovered using the inverse processing of the PSS, which can be performed by Lagrange polynomials.[11] This process can recover all secret files $\{F_1, \ldots, F_f\}$.

5 Experimental Results

Two input binary secret images and two input secret files are shown in Figs. 5 and 6, respectively. In our experiment, the

(a) (b)

Fig. 5 Two input binary secret images $\{S_1, S_2\}$ whose decoding is through human eyes only. (a) A 256×256 image book and (b) a 192×128 image key.

(a) (b)

Fig. 6 Two input secret files whose decoding requires a computer. (a) A compressed 512×512 gray-level secret image Lena, and its file size is 14.9 KB and (b) a compressed 512×512 gray-level image boat, and its file size is 15.4 KB.

two 256×256 transparencies shown in Figs. 7(a) and 7(b) are generated by applying the translated FVC scheme in Sec. 3 to the two binary images of book and key in Fig. 5. Then, two new large transparencies of 512×512 each are generated in Sec. 4, as shown in Figs. 7(e) and 7(f), by combining Figs. 7(a) and 7(b) and the information of gray value images in Figs. 6(a) and 6(b). Then, we obtain two VC decoding results to decode the two "same size" transparencies (both are 512×512 or both are 256×256). One result is stacking the same-size transparencies directly. Figure 7(c) shows the decoding result by stacking Figs. 7(a) and 7(b) together. Likewise, Fig. 7(g) shows the decoding result by stacking Figs. 7(e) and 7(f) together. The other VC decoding result is through flipping and translation. Figure 7(d) shows the decoding result by stacking Fig. 7(a) with Fig. 7(b). Before performing this stacking, we flipped Fig. 7(a) and applied translation to Fig. 7(b), namely, 61 pixels horizontally and 125 pixels vertically. Likewise, Fig. 7(h) shows the decoding result by stacking Fig. 7(e) with Fig. 7(f). Before stacking, we flipped Fig. 7(e) and applied translation to the transparency in Fig. 7(f), namely, 61×2 pixels horizontally and 125×2 pixels vertically. In Figs. 7(d) and 7(h), the translation of the pixel position should be precise. Otherwise, the stacking will still provide a noise-like result.

Finally, the computer decoding option is: (1) to extract the information hidden in the transparencies shown in Figs. 7(e) and 7(f); (2) then, we can obtain the gray-valued secret files (Lena and boat) of Fig. 6 identically by performing inverse PSS using a computer and the information recently extracted from Figs. 7(e) and 7(f).

6 Comparison and Analysis

6.1 *Comparison*

The shift operation used in papers[3–5] is similar to the translated operation used in our translated FVC method in Sec. 3. However, their pixel expansion rate is $m = 4$, and thus four times larger than $m = 1$ of our translated FVC method in Sec. 3. The translated FVC method in Sec. 3 exhibits no pixel expansion. The contrast is $1/6$. As stated in the FVC method in Ref. 6, $1/6$ is the optimal contrast if people require no pixel expansion and no secret leaking.

Table 2 lists the comparison between our method and some well-known reported methods. Reference 19 introduced the VCPSS method, which also possesses two-decoding options (VC and PSS double decoding). However, it has neither the translation function nor the flipping function to decode the secret image. Thus, the number of their binary image is one, whereas we decrypt two binary images. Both Refs. 2 and 6 support multiple secrets. However, both exhibited only one decoding option, that is, VC decoding. References 11 and 20 utilized a PSS scheme. Their size of shares can even be smaller than the size of the secret file. However, a computer is always required because they do not possess human-vision instant decoding. The elegant method in Ref. 21 hides one binary secret image. However, it can provide a consistent and positive contrast for different revealed regions while the scheme involves a large pixel expansion rate m. Reference 22 embedded two-dimensional (2-D) barcodes into the VC shares for authentication to prevent cheating. The barcodes can be detected by scanners and might be perceived by human eyes. In contrast, our secret files are totally invisible on transparencies. Our method demonstrates all-in-one two decoding options. Two binary secret images can be revealed by stacking, flipping, and translation, whereas secret files, which might be fine gray-level images or authenticated certificate such as SHA-256, can be decoded by a computer.

6.2 *Analysis on the Visual Effect Perseverance of the 2×2 Expansion*

When we expand transparencies from $\{T_1, T_2\}$ to two large transparencies $\{T_1', T_2'\}$, we aim for a stacking result using $\{T_1', T_2'\}$, which is similar to the stacking result using $\{T_1, T_2\}$. To achieve this, we require a black pixel in the stacking result $T_1 \vee T_2$ that will relate to a 2×2 expansion, which looks darker than the 2×2 expansion of a white pixel in stacking result $T_1 \vee T_2$. Below, we show that this requirement is satisfied.

Table 3 presents the statistical results of stacking two 2×2 blocks coming from the set $\{B_0, B_1, B_2, B_3, B_4\}$ or $\{W_0, W_1, W_2, W_3, W_4\}$. If at least one of the two blocks is of black-block type $\{B_0, \ldots, B_4\}$, then the stacking result is a 2×2 block that is expected to be nontransparent in most of its four elements because

$$(B \vee B) = (B \vee W) = (W \vee B) = B$$

when we stack transparencies (as W indicates transparence, i.e., limpidity). Then, the possible situations after stacking are

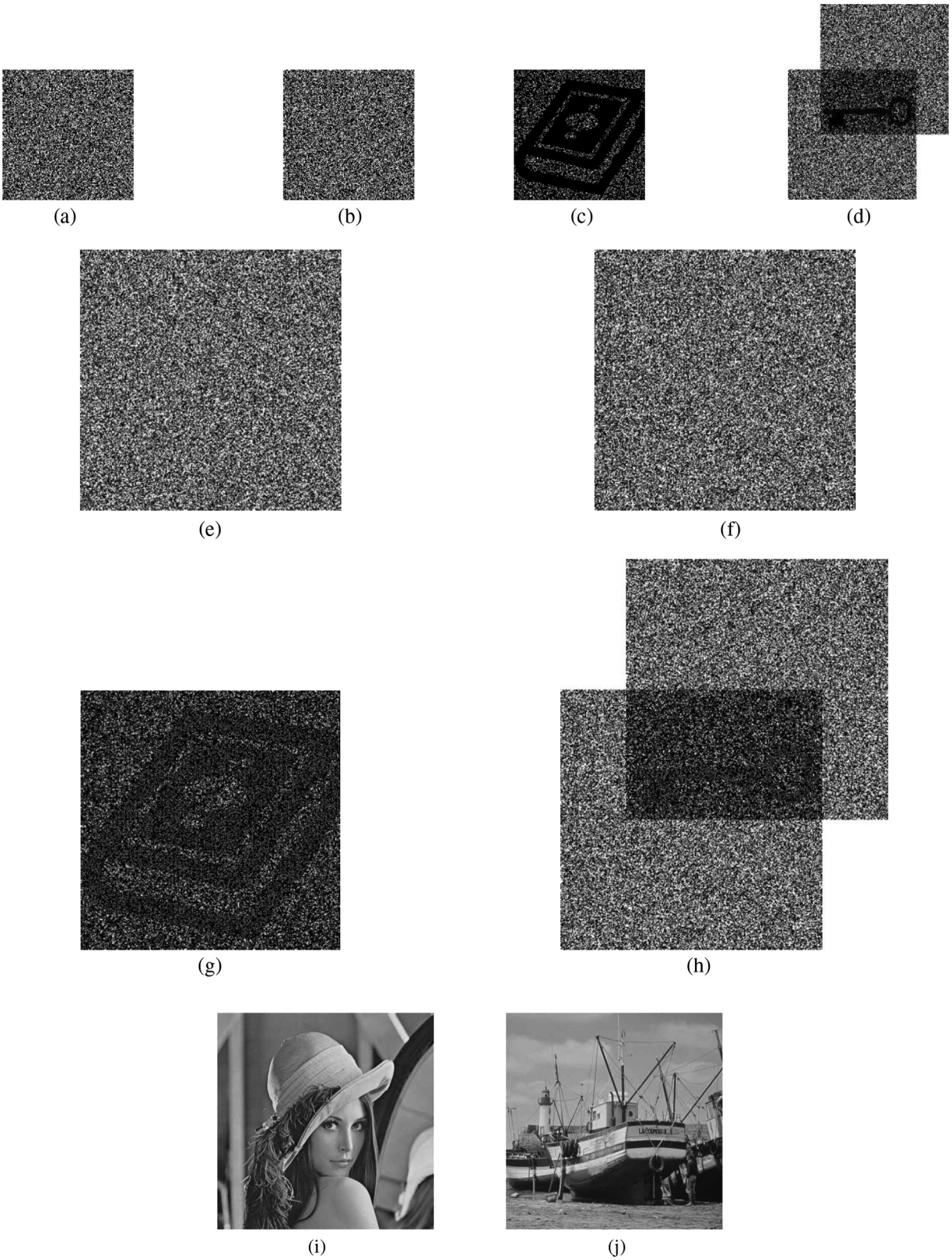

Fig. 7 Results of our experiment. (a, b) are the two 256×256 transparencies created in Sec. 3; (c, d) are the stacking results of (a) and (b); (e, f) are the two 512×512 output transparencies, which are generated by combining (a) and (b) and the images in Figs. 6(a) and 6(b); (g, h) are the stacking results of (e) and (f); (i, j) are the recovered gray-level images using a computer and (e) and (f).

Table 2 Comparison with some reported methods.

Methods	Character (VC or PSS)	Number of secret	Pixel expansion rate m (prefer small)	Contrast (prefer large)	Both visual and computer decoding	Instant decode without computer	Use translation and flip in VC decoding
Method[19]	VC + PSS	One binary image, one gray image	$(k=2,n=4)$:4[a,b] $(k=3,n=4)$:6[a,b]	$(k=2,n=4)$:1/5 to 1/6[a] $(k=3,n=4)$:1/7 to 1/9[a]	Yes	Yes	None
Method[6]	VC only	Two binary images	1, i.e., no expansion	1/6	No	Yes	Flip only
Method[2]	VC only	One binary image and one text file	4	1/2	No	Yes	Translation only
Methods[11,20]	PSS only	One gray image	Shares can be smaller than secret	—	No	No	None
Method[21]	Region-Increm. VC	One binary image	$(k=2,n=4)$:18[a,b] $(k=3,n=4)$:16[a,b]	$(k=2,n=4)$:1/18[a,b] $(k=3,n=4)$:1/16[a,b]	No	Yes	None
Method[22]	VC + scan	One binary image and 2-D barcodes	Depends on choice of VC methods	Depends on choice of VC methods	Yes	Yes	None
Our translated FVC (Sec. 3)	VC only	Two binary images	1, i.e., no expansion	1/6	No	Yes	Both translation and flip
Our T-FVC-PSS (Sec. 4)	VC + PSS	Two binary images and some secret files	4	<1/6 after expansion	Yes	Yes	Both translation and flip

[a]value can be influenced by the usage of different (k,n) schemes.
[b]A direct quotation from the reported study.

Table 3 Statistics about results of stacking two 2×2 blocks using our five types of blocks.

	Block type $\in \{B_0, B_1, B_2, B_3, B_4\}$	Block type $\in \{W_0, W_1, W_2, W_3, W_4\}$
Block type $\in \{B_0, B_1, B_2, B_3, B_4\}$	All four pixels are black: 21 cases	All four pixels are black: 9 cases
	Three black pixels: 4 cases	Three black pixels: 16 cases
Block type $\in \{W_0, W_1, W_2, W_3, W_4\}$	All four pixels are black: 9 cases	All four pixels are white: 1 case ($W_0 \vee W_0$)
	Three black pixels: 16 cases	Three white pixels: 12 cases
		Two white pixels: 12 cases

(1) If the two blocks being stacked are both from the set $\{B_0, B_1, B_2, B_3, B_4\}$, then $5\times5=25$ possible combinations $\{B_0 \vee B_0, \ldots, B_4 \vee B_4\}$ exist. In the 25 combinations, four combinations $\{B_1 \vee B_1, B_2 \vee B_2, B_3 \vee B_3$, and $B_4 \vee B_4\}$ exist wherein the 2×2 stacking result exhibits three black pixels. The remaining 21 combinations yield a stacking result whose $2\times2=4$ pixels are all black.

(2)

(a) If the first of the two blocks being stacked is from the set $\{B_0, B_1, B_2, B_3, B_4\}$ and the second block is from the set $\{W_0, W_1, W_2, W_3, W_4\}$, then, in the 25 possible combinations, nine combinations ($B_0 \vee \{W_0, W_1, W_2, W_3, W_4\}$; $B_1 \vee W_1$, $B_2 \vee W_2$, $B_3 \vee W_3$, and $B_4 \vee W_4$) yield a stacking result whose $2\times2=4$ pixels are all black pixels. The remaining 16 combinations yield stacking results whose $2\times2=4$ pixels are formed by three black pixels and one white pixel.

(b) If the first block is from $\{W_0, W_1, W_2, W_3, W_4\}$ and the second block is from $\{B_0, B_1, B_2, B_3, B_4\}$, then the statistics of the stacking result is identical to that of (a). In other words, nine combinations generate stacking results whose $2\times2=4$ pixels are all black, and 16 combinations generate stacking results whose $2\times2=4$ pixels are formed of three blacks and one white.

(3) If the two blocks being stacked are both from the set $\{W_0, W_1, W_2, W_3, W_4\}$, then in the $5\times5=25$ possible combinations, only one combination $\{W_0 \vee W_0\}$ exists whose $2\times2=4$ pixels of the stacking result are all white pixels. Twelve combinations $\{W_0 \vee \{W_1, W_2, W_3, W_4\}$; $W_1 \vee \{W_0, W_1\}$; $W_2 \vee \{W_0, W_1\}$; $W_3 \vee \{W_0, W_1\}$; and $W_4 \vee \{W_0, W_1\}\}$ result in having one black pixel. The remaining 12 combinations result in having two black pixels.

In the black blocks expected in the decoding result after stacking, a $(21+9+9)/(25+25+25)=52\%$ chance exists that all four pixels of the block are black pixels and a $(4+16+16)/(25+25+25)=48\%$ chance exists that three of the four pixels of the block are black pixels. Meanwhile, for the white blocks, a $1/25=4\%$ chance exists that all four pixels of the block are white pixels. In addition, a $12/25=48\%$ chance exists that three of the four pixels of the block are white pixels, and a $12/25=48\%$ chance exists that two of the four pixels of the block are white. The above probability distribution means that: (1) 52% of the black area are all black, and 48% of the black area are majorly black and (2) 48% of the white area are gray, 48% of the white area are majorly white, and 4% of the white area are all white. This is good enough for human vision to distinguish the black area from the white area, and thus identify images because a black block has more black pixels.

In our design, each block has five types. To hide a byte, we need five blocks because $5^4 \geq 256$. If we use more than five types, then we can hide more data, but the visual quality of the stacking results will get worse, as explained below. For example, if we use seven types of blocks, then we need only three blocks to hide a byte, because $7^3 \geq 256$. However, using seven types of blocks not only yields a lower contrast (hard to distinguish the white area from the black area) but also makes a contradiction to human vision (the white area looks darker than the black area) as analyzed below. In addition to the five types $\{B_0, B_1, B_2, B_3, B_4\}$ and $\{W_0, W_1, W_2, W_3, W_4\}$ defined in Eqs. (13) and (14), which meet the requirements stated in Eqs. (11) and (12), if we want to introduce two more types, then the four binary values $[a_0\, a_1\, a_2\, a_3]$ of each 2×2 bock in $\{B_5, B_6\}$ and $\{W_5, W_6\}$ must have $a_0+a_1+a_2+a_3=2$, for Eqs. (13) and (14) indicates that the case $a_0+a_1+a_2+a_3=4$, or 3, or 1, or 0 must yield a block in $\{B_0, B_1, B_2, B_3, B_4; W_0, W_1, W_2, W_3, W_4\}$. Since $a_0+a_1+a_2+a_3=2$, the four pixels of the 2×2 block must be two white pixels and two black pixels. Without the loss of generality, we use

$B_5 =$  , $B_6 =$  , $W_5 =$  , and $W_6 =$ 

to illustrate. Table 4 presents the statistical results if we stack two 2×2 blocks coming from the set $\{B_0, B_1, B_2, B_3, B_4, B_5, B_6\}$ and $\{W_0, W_1, W_2, W_3, W_4, W_5, W_6\}$. For black blocks, a $(35+15+15)/(49+49+49)=44.2\%$ chance exists that all four pixels of the block are black, a $(12+28+28)/(49+49+49)=46.3\%$ chance exists that three of the four pixels are black, a $(2+6+6)/(49+49+49)=9.5\%$ chance exists that two of the four pixels are black. For white blocks, a $1/49=2\%$ chance exists that all four pixels of the block are white, a $12/49=24.5\%$ chance exists that one of the four pixels are black, a $26/49=53.1\%$ chance exists that two of the four pixels are black, and a $10/49=20.4\%$ chance exists that three of the four pixels are black. The above analysis shows that, after stacking, some white blocks

Table 4 Statistics about results of stacking two 2×2 blocks using seven types of blocks.

	Block type $\in \{B_0, B_1, B_2, B_3, B_4, B_5, B_6\}$	Block type $\in \{W_0, W_1, W_2, W_3, W_4, W_5, W_6\}$
Block type $\in \{B_0, B_1, B_2, B_3, B_4, B_5, B_6\}$	All four pixels are black: 35 cases	All four pixels are black: 15 cases
	Three black pixels: 12 cases	Three black pixels: 28 cases
	Two black pixels: 2 cases	Two black pixels: 6 cases
Block type $\in \{W_0, W_1, W_2, W_3, W_4, W_5, W_6\}$	All four pixels are black: 15 cases	All four pixels are white: 1 case
	Three black pixels: 28 cases	Three white pixels: 12 cases
	Two black pixels: 6 cases	Two white pixels: 26 cases
		One white pixel: 10 cases

even look darker than some black blocks; this kind of mistake will make the secret image not able to be unveiled by human vision. Therefore, we cannot use seven types. The analysis of avoiding the use of six types is similar to the above analysis.

6.3 *Hiding Capacity*

Let $|F_{\text{sum}}|$ be the total number of bytes of the f secret files $\{F_1, F_2, \ldots, F_f\}$. Then, with a $(n = 2,\ k = 2)$ PSS, each shadow has $|F_{\text{sum}}|/2$ bytes. After transforming the base from 256 to 5, the number of base five digits of a shadow is

$$(|F_{\text{sum}}|/2) \times (\log 256/\log 5) = (|F_{\text{sum}}|/2) \times 3.445 \leq (|F_{\text{sum}}|/2) \times 4 = 2|F_{\text{sum}}|. \quad (18)$$

Now, if the large dimension of the two input binary images for VC encryption is $N \times M$, then the two VC transparencies created in Sec. 3 are also $N \times M$ each, and the two expanded transparencies created in Sec. 4 are $2N \times 2M$ each. Each expanded transparency possesses NM blocks of 2×2 each. Each 2×2 block can hide a digit of base 5. Thus, NM blocks can hide NM digits. Therefore, an assumption is derived based on Eq. (18) as follows:

$$1.7225|F_{\text{sum}}| \leq 2|F_{\text{sum}}| \leq NM,$$

which can ensure that the hiding space is sufficient if we hide all secret files using the five types of the transparency block. For example, $N \times M = 256 \times 256$ in our experiment shown in Fig. 7 in Sec. 5. Thus, the f secret files together should be of $|F_{\text{sum}}|$ bytes with $|F_{\text{sum}}| \leq NM/2 = 256 \times 256/2 = 32{,}768$ (or, with $|F_{\text{sum}}| \leq NM/1.7225 = 256 \times 256/1.7225 = 38{,}047$ bytes if in the hiding rather than transform every byte independently from base 256 to base 5 in a byte-by-byte manner, and we treat the whole shadow as a very large number and transform it from base 256 to base 5).

6.4 *Security Analysis*

Three kinds of security levels are provided here to protect distinct secrets. Binary secret image S_1 is of the lowest security level. Its protection is similar to most of the reported VC methods based on the separate storage of the two generated transparencies. People cannot decode S_1 unless they obtain all transparencies. Binary secret image S_2 is a little harder to decode than S_1 because of the use of translation and flipping. In addition, the major protection of S_2 is based on the separate storage of the generated transparencies. Thus, we use the storage separation of the two transparencies to protect two binary secret images.

As for the combination of our method with PSS, our shadows are created by a (2, 2) PSS. If a thief obtains only a transparency, he will be unable to reveal any secret file in $\{F_1, F_2, \ldots, F_f\}$ (for example, gray-value images) because of the use of the PSS scheme. In addition, we encrypt shadows through the LCG algorithm. This method provides a secure environment. In Eq. (16), $P = 2^x$, $x > 2$ is assumed. If we want the LCG algorithm to exhibit a full period, then $2^{x-2} - 1$ possibilities exist for A, $2^{x-1} - 1$ possibilities for B, and $2^x - 1$ possibilities for X_0. Overall, almost $2^{3(x-1)}$ possible combinations of $(A,\ B,\ X_0)$ exist. For example, if $P = 256 \times 256 = 2^{16}$, then almost $2^{3(16-1)} = 2^{45}$ possible combinations of $(A,\ B,\ X_0)$ exist. Therefore, the protection of the secret files $\{F_1, F_2, \ldots, F_f\}$ (for example, gray-value image) is triply guarded by: (1) the storage separation of the two transparencies, (2) the $(A,\ B,\ X_0)$ key to encrypt the shadows using the LCG algorithm or the keys by other encryption algorithms, and (3) the PSS parameter that we use to create the shares. For example, we can create six shares. Each share is $1/6$ of the original file size rather than creating two shares that are $1/2$ of the original file size. Then, distinct binding of the six shares into two groups will cause a distinct decoding strategy. The $(6,\ 1/6)$ can be replaced by other $(k,\ 1/k)$ values.

6.5 *Translated FVC versus T-FVC-PSS*

In summary, if it is expected that no computer is available in the decoding end, then we suggest the use of translated FVC. If a user is quite concerned about the security of a true secret, then we suggest the use of T-FVC-PSS. The details are explained below.

When a computer is not available, translated FVC provides us a handy way to share our secrets via two transparencies. Each transparency has no information leak and has no pixel expansion. The contrast value of translated FVC is better than T-FVC-PSS, so the revealed secrets would be more easily recognized. However, T-FVC-PSS is more unbreakable than translated FVC if a thief is lucky enough to obtain both transparencies from two distinct places. He can view the

binary secret image S_1 immediately; however, unveiling the second secret S_2 entails many attempts to obtain a suitable translation-flip stacking. As explained in Sec. 6.6, we may use two fake images as S_1 and S_2 to mislead the thief into thinking that he has already discovered everything. Because the true secret S_3 can be decoded only by using a totally different way (i.e., by a computer), which requires the knowledge of many unknown PSS parameter values and LCG keys (A, B, X_0) or keys of other encryption tools alternative to LCG.

6.6 *Level of Confusion Protection*

To protect a true secret against an intruder who steals both transparencies, our method in Sec. 4 provides strong confusion protection (the intruder might think that no further secret exists). Notably, the elegant recursive VC schemes proposed by Katta in Ref. 9 and Kumar and Sharma in Ref. 10 can hide many secret binary images. Since so many secret images can be hidden, these systems can have confusion ability too. However, by the observation below, we find that their confusion protection of the true secret is different and not as strong as ours, for the intruder can still view the true secret. In Refs. 9 and 10, every secret is decoded by stacking transparencies using a specified translation. In particular, their translation amount is a fixed fraction of the transparency size (using Ref. 10 as an example, the first shift is half of the transparency width; then the second shift is half of the transparency height; then the third shift is one-fourth of the transparency width; then the fourth shift is one-fourth of the transparency height; and each of the shifts above unveils a secret image). Since the recursive shift amounts are very regular, it is not hard for the intruder to decode all the hidden secret images. Therefore, the confusion protection against the intruder is not as strong as ours. In our system, decoding second image S_2 requires flipping together with a translation whose translation amount is quite free and harder to guess. Decoding true secret image S_3 even needs a computer rather than stacking. In other words, each image of ours has its own "method" to decode (rather than its own "translation amount" to decode). This gives stronger confusion for an intruder; especially in the jump from stacking to computer, when the two fake images are decoded by VC and the true one is decoded by computer.

Our other advantages over theirs are: (1) our third secret S_3, which is always a true secret, can be any secret file not limited to binary, gray, or color images; (2) our true secret S_3 can be very large; even larger than the largest fake binary secret S_1. In our experiment, if the largest fake binary secret S_1 is $(256 \times 256/8)/1024 = 8$ K bytes, then the true secret S_3 can be as large as 32 K bytes. In the recursive systems,[9,10] the image with the largest size is easiest to decode, for its decoding is a direct stacking without any shift. So, for the purpose of protection, the largest image should not be a true image. The next six decoded secrets are, respectively, q, q^2, q^3, q^4, q^5, and q^6 times smaller than the largest secret, where $q \geq 3$ in Ref. 9 and $q = 2$ in Ref. 10. Hence, if their recursive hiding systems are used, then their true secret is very small; regardless if it is compared to the fake secret S_1 of largest size, or compared to our true secret.

Of course, the recursive systems[9,10] have an advantage: many secrets (from large size to small size) can be embedded. With recursive translations, people can view "all" secrets (including true secrets) by stacking transparencies in a regular manner. Then, the intruder needs to judge which unveiled secrets are true and which are fake. But this kind of "mixing together" protection is different from our "concealing the true secret" protection. In our policy, we just completely erase the chance of viewing the true secret S_3 using stacking (although the two fake secrets are viewed using stacking). As a remark, even though the confusion ability of Refs. 9 and 10 is not as strong as ours, the readers should notice that their goal of design is to hide as many secrets as possible, rather than giving confusion.

7 Summary

This study presents an all-in-one scheme with both visual and computer decodings. First, as a preprocessing, we propose a translated FVC method in Sec. 3. Binary secret image S_1 can be viewed by stacking two transparencies together. The other binary secret image S_2 can be viewed by flipping the first transparency and translating the second transparency before stacking. Translated FVC provides people an alternative way to hide binary secret images. This method is without pixel expansion. In Sec. 4, we enhance the translated FVC in Sec. 3 with a computer decoding option. The enhanced method is called T-FVC-PSS, which preserves all functionality of the translated FVC while hiding in the transparencies more secret files (for example, gray-value images or color images or military plans) which are larger than S_1 and S_2. The two transparencies in Sec. 4 can not only show the two binary secret images S_1 and S_2 by stacking, flipping, and translation but can also decode several secret files by a computer. Thus, different secrets require their own decoding processes. It is a new alternative to image hiding because of the use of transparencies as host media to replace the role of host images when fine secret files, such as military plans or true color images, need to be protected. The double decoding manners with/without a computer, together with the use of transparency media, also give a true secret more protection.

Acknowledgments

The author would like to thank the reviewers for the valuable suggestions. This work was supported by the Ministry of Science and Technology, R.O.C., under the grant MOST103-2221-E-009-119-MY3.

References

1. M. Naor and A. Shamir, "Visual cryptography," in *Proc. of Advances in Cryptology EUROCRYPT 1994*, Perugia, pp. 1–12 (1994).
2. W. P. Fang and J. C. Lin, "Visual cryptography with extra ability of hiding confidential data," *J. Electron. Imaging* **15**(2), 023020 (2006)
3. T. H. Chen, K. H. Tsao, and C. S. Wu, "Multi-secrets visual secret sharing," in *2008 Asia-Pacific Conf. on Communications (APCC)*, Tokyo, pp. 1–5 (2008).
4. D. Wang et al., "Shift visual cryptography scheme of two secret images," *Prog. Nat. Sci.* **13**(6), 457–463 (2003).
5. H. Luo et al., "Watermarking-based transparency authentication in visual cryptography," in *Seventh Int. Conf. on Intelligent Systems Design and Applications (ISDA)*, Brazil, pp. 609–616 (2007).
6. S. J. Lin, S. K. Chen, and J. C. Lin, "Flip visual cryptography (FVC) with perfect security, conditionally-optimal contrast, and no expansion," *J. Vis. Commun. Image Represent.* **21**(8), 900–916 (2010).
7. S. J. Shyu et al., "Sharing multiple secrets in visual cryptography," *Pattern Recognit.* **40**(12), 3633–3651 (2007).
8. S. J. Shyu and K. Chen, "Visual multiple secret sharing based upon turning and flipping," *Inf. Sci.* **181**(15), 3246–3266 (2011).
9. S. Katta, "Recursive information hiding in visual cryptography," Cryptology ePrint Archive, Report 2010/283, p. 283 (2010).
10. S. Kumar and R. K. Sharma, "Recursive information hiding of secrets by random grids," *Cryptologia* **37**(2), 154–161 (2013).

11. C. C. Thien and J. C. Lin, "Secret image sharing," *Comput. Graphics* **26**(5), 765–770 (2002).
12. S. K. Chen, "Essential secret image sharing with increasable shadows," *Opt. Eng.* **55**(1), 013103 (2016).
13. M. Ulutas, G. Ulutas, and V. V. Nabiyev, "Medical image security and EPR hiding using Shamir's secret sharing scheme," *J. Syst. Softw.* **84**(3), 341–353 (2011).
14. W. K. Chen and H. K. Tso, "Visual sharing protection method for medical images," *J. Med. Syst.* **37**(1), 1–8 (2013).
15. E. Elsheh and A. B. Hamza, "Secret sharing approaches for 3D object encryption," *Expert Syst. Appl.* **38**(11), 13906 (2011).
16. S. S. Lee, Y. J. Huang, and J. C. Lin, "Protection of 3D models using cross recovery," *Multimedia Tools Appl.* **76**(1), 243–264 (2017).
17. V. Kumar and R. Kumar, "Detection of phishing attack using visual cryptography in ad hoc network," in *2015 Int. Conf. on Communications and Signal Processing (ICCSP)*, Tamilnadu, pp. 1021–1025 (2015).
18. S. Y. Chang et al., "Progressive sharing of multiple images with sensitivity-controlled decoding," *EURASIP J. Adv. Signal Process.* **2015**(1), 1–19 (2015).
19. S. J. Lin and J. C. Lin, "VCPSS: a two-in-one two-decoding-options image sharing method combining visual cryptography (VC) and polynomial-style sharing (PSS) approaches," *Pattern Recognit.* **40**(12), 3652–3666 (2007).
20. A. Nag et al., "A new (k, n) verifiable secret image sharing scheme (VSISS)," *Egypt. Inf. J.* **15**(3), 201–209 (2014).
21. S. Li, J. Li, and D. Wang, "Region incrementing visual cryptography scheme with same contrast," *Chin. J. Electron.* **25**(4), 621–624 (2016).
22. G. Wang, F. Liu, and W. Q. Yan, "2D barcodes for visual cryptography," *Multimedia Tools Appl.* **75**(2), 1223–1241 (2016).

Chia-Hua Wu received her BS degree in computer science and engineering in 2012 from National Sun Yat-sen University and her MS degree in computer science from National Chiao Tung University in 2014. Currently, she is a software engineer at VIA Technologies, Inc.

Suiang-Shyan Lee received his BS degree in computer science and information engineering in 2007 and his MS degree in computer science in 2009. Currently, he is a PhD candidate in the Department of Computer Science at National Chiao Tung University, Taiwan, and an engineer at QNAP Systems, Inc. His research interests include multimedia security and computer graphics.

Ja-Chen Lin received BS and MS degrees from National Chiao Tung University, Taiwan, and a PhD in mathematics from Purdue University, USA. He was a graduate instructor at Purdue University from 1984 to 1988. He joined the Department of Computer Science at National Chiao Tung University, in August 1988, where he is currently a professor. His current research interests include pattern recognition and image processing.