Available online at www.sciencedirect.com
ScienceDirect
Pattern Recognition 40 (2007) 3652–3666
PATTERN RECOGNITION
THE JOURNAL OF THE PATTERN RECOGNITION SOCIETY
www.elsevier.com/locate/pr

VCPSS: A two-in-one two-decoding-options image sharing method combining visual cryptography (VC) and polynomial-style sharing (PSS) approaches

Sian-Jheng Lin, Ja-Chen Lin*

Department of Computer Science, National Chiao Tung University, Hsinchu, Taiwan, ROC

Received 17 January 2006; received in revised form 4 April 2007; accepted 5 April 2007

Abstract

This paper presents a novel method to combine two major branches of image sharing: VC and PSS. n transparencies are created for a given gray-valued secret image. If the decoding computer is temporarily not available at (or, not connected to) the decoding scene, we can still physically stack any t received transparencies ($t \leqslant n$ is a threshold value) to get a vague black-and-white view of the secret image immediately. On the other hand, when the decoding computer is finally available, then we can get a much finer gray-valued view of the secret image using the information hidden in the transparencies. In summary, each transparency is a two-in-one carrier of the information, and the decoding has two options.

Keywords: Hiding using block-types; Image sharing; (t, n) threshold scheme; Halftone version; Compression

1. Introduction and goal

Image sharing can be used in a team when no member alone should be trusted. Visual cryptography (VC) [1–7] and polynomial-style sharing (PSS) [8–14] are both well-known branches to share images. Both can be designed as (t, n) schemes. (In this paper, we say that a sharing technique is (t, n) if and only if it shares a secret image $\boldsymbol{S}$ among n shadows so that any t of the n shadows ($n \geqslant t$) can unveil the secret image $\boldsymbol{S}$ (or a compressed version $\boldsymbol{S}^{(\mathrm{comp})}$ of $\boldsymbol{S}$), whereas less than t shadows cannot.) Although both VC and PSS can share images, they are quite different in many manners. Table 1 below compares VC and PSS.

In Table 1, if we temporarily ignore the final column (which is for the future comparison use in the experiment section, we list this column here just to save paper's space), we can see that VC is simple and fast, while PSS gives good image quality. A question arises naturally: "Can VC be combined with PSS?" To certain extent, the answer is positive, as is shown here. In this paper, we present a method to combine these two techniques and achieve a goal: if the decoding computer is temporarily not available in (or, not connected to) the decoding scene, we can still physically stack the t received shadows to get a vague black-and-white view of the secret image immediately; later, when the computer is finally available, we can get a much finer gray-valued view of the secret image using the information hidden earlier in the shadows by using PSS. (Hereinafter, the final output of the proposed method will be called as "transparencies" rather than "shadows" because, as mentioned above, one of the two decoding manners is that the shadows can be stacked physically for viewing, just like ordinary transparencies can be stacked and viewed.)

Below in Section 2 we first review some background knowledge used in this paper, and then in Section 3 we introduce our method. The experimental results are in Section 4, and the conclusions are in Section 5. Section 6 describes an application of this paper.

2. Background

Some background knowledge is reviewed in this section. Section 2.1 reviews the basis matrices roughly, which is a

* Corresponding author. Fax: +886 3 5721490.
E-mail address: jclin@cis.nctu.edu.tw (J.-C. Lin).

0031-3203/$30.00
doi:10.1016/j.patcog.2007.04.001

Table 1
A comparison between VC and PSS

	Visual cryptography (VC, see [1–7])	Polynomial-style sharing (PSS, see [8–14])	Ours (VC + PSS)
Usually, the input secret image $\boldsymbol{S}$ is	Black-and-white	Gray	Gray
Decoding speed (and decoding method)	Instant (by using eyes after stacking shadows)	Slow (by computation)	Instant in Layer 1; slow in Layer 2.
Is a computer needed in decoding?	No	Yes	"No" in Layer 1; "yes" in Layer 2.
Recovered image's perceptual quality	Vague	Fine	Vague in Layer 1; fine in Layer 2.
Size of each shadow	Larger than that of $\boldsymbol{S}$	Can be smaller than that of $\boldsymbol{S}$	Either the length or the width of a (binary) transparency is larger than that of $\boldsymbol{S}$, but the number of computer-storage bytes needed can be smaller than $\boldsymbol{S}$'s.

background knowledge well known in VC field; Section 2.2 reviews the PSS technologies, including Shamir's [14] and Thien and Lin's [8].

2.1. A review of the basis matrices $[\mathbf{B_0}]$ and $[\mathbf{B_1}]$ for VC

Below we review the two basis matrices $[\mathbf{B_0}]$ and $[\mathbf{B_1}]$ often mentioned in VC field (e.g. see Ref. [1]). The matrix $\mathbf{B_0}$ is called a "white matrix" because it is useful to produce blocks whose stacking result will represent white pixels of a black-and-white (e.g. halftone) image. Matrix $[\mathbf{B_1}]$ is called a "black matrix" for analogous reason. Without the loss of generality, below we only show the case $(t, n) = (2, 4)$, i.e. only two out of four shares are needed in recovering. For a general pair of given values (t, n), the readers may either design their own $[\mathbf{B_0}]$ and $[\mathbf{B_1}]$, or use the Appendix to create some pairs of $[\mathbf{B_0}]$ and $[\mathbf{B_1}]$. In fact, even if the values of t and n are fixed, the choice of $[\mathbf{B_0}]$ and $[\mathbf{B_1}]$ is still not unique. To apply the proposed VCPSS two-in-one sharing method, people can use any pair of $[\mathbf{B_0}]$ and $[\mathbf{B_1}]$ satisfying the requirements (i)–(iii) stated in next paragraph (these three requirements also appear in the Appendix). In summary, the pair $[\mathbf{B_0}]$ and $[\mathbf{B_1}]$ is not necessarily generated from the Appendix; the Appendix is just to let readers know that there always exists at least one solution to find out $[\mathbf{B_0}]$ and $[\mathbf{B_1}]$.

In the $(t, n) = (2, 4)$ case, one of the several possible choices for the white matrix $[\mathbf{B_0}]$ and the black matrix $[\mathbf{B_1}]$ is to use

$$[\mathbf{B_0}] = \begin{bmatrix} 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 \end{bmatrix} \quad \text{and} \quad [\mathbf{B_1}] = \begin{bmatrix} 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{bmatrix}. \tag{1}$$

Both matrices have $n = 4$ rows. (In general, no matter how we assign the two matrices, each matrix must have n rows if n transparencies are to be created. This is the so-called requirement (i).) In both matrices, each 0 means that a white element is painted there, and each 1 means a black element is painted there. As we can see, both $[\mathbf{B_0}]$ and $[\mathbf{B_1}]$ have two black elements per row. (In general, the number of 1's appearing in each row of $[\mathbf{B_0}]$ must be identical to that of $[\mathbf{B_1}]$. This is the so-called requirement (ii).) It is also obvious that if we stack any two $(=t)$ rows of our $[\mathbf{B_0}]$, the stacking result has two black elements and two white elements. On the other hand, if we stack any two $(=t)$ rows of $[\mathbf{B_1}]$, the stacking result has at least three black elements. (In general, no matter how we choose $[\mathbf{B_0}]$ and $[\mathbf{B_1}]$, the number of 1's contained in the result of stacking any t rows of $[\mathbf{B_1}]$ must exceed that of stacking any t rows of $[\mathbf{B_0}]$. This is the so-called requirement (iii).)

Now, assume that we want to create $4(=n)$ blocks, each is 2×2 in size, so that stacking any $2(=t)$ of them will yield a 2×2 so-called "white block" (defined here as a block in which only two of the four elements are 1's (i.e. only two black elements)). All we have to do is to permute the columns of $[\mathbf{B_0}]$ randomly, and then distribute the $4(=n)$ rows of the permuted $[\mathbf{B_0}]$ to four customers. After that, each customer uses the first two elements as the first row of his block, and next two elements as the 2nd row of his block. As a result, each of the $4(=n)$ customers has his own 2×2 block, and any two of these four 2×2 blocks can be stacked to yield a 2×2 white block (only two of its $2 \times 2 = 4$ elements are 1's).

Similarly, if we want that any $t(=2)$ of the $n(=4)$ created blocks (each is still 2×2 in size) can be stacked to yield a so-called "black block" (defined as a 2×2) block in which at least three of the four elements are 1's (i.e. at least three black elements), then we only have to replace the role of $[\mathbf{B_0}]$ by $[\mathbf{B_1}]$ in the above argument and obtain four blocks corresponding to $[\mathbf{B_1}]$. Then distribute these four blocks arbitrarily to the four customers (one block per customer).

In the above example, each block has $w = 2$ white elements and $b = 2$ black elements, (or equivalently, each row of $[\mathbf{B_0}]$ or $[\mathbf{B_1}]$ has two white elements and two black elements), and the permutation of the columns of $[\mathbf{B_0}]$ or $[\mathbf{B_1}]$ will not affect the stacking result's brightness (i.e. number of black elements of