Hiding Authenticable General Digital Information behind Binary Images with Reduced Distortion

Chih-Hsuan Tzeng
Department of Computer and Information Science,
National Chiao Tung University
Hsinchu, Taiwan 300, Republic of China
Tel: +886-3-5728368
chtzeng@cis.nctu.edu.tw

Wen-Hsiang Tsai
Department of Computer and Information Science,
National Chiao Tung University
Hsinchu, Taiwan 300, Republic of China
Tel: +886-3-5728368
whtsai@cis.nctu.edu.tw

ABSTRACT

A new approach to information hiding in binary images with the capabilities of hidden data authentication and image distortion reduction is proposed. The hidden information may be of any data form. Based on a new feature called surrounding edge count for measuring the structural randomness in an image block, pixel embeddability is defined from the viewpoint of minimizing image distortion. Accordingly, embeddable image pixels suitable for hiding secret data are selected. Furthermore, an error-correcting scheme is used both for hidden data authentication and for image distortion reduction. Finally, to increase the security of embedded data, a secret key and a random number generator are employed to randomize the locations of selected pixels into which secret data are embedded. Experimental results show the feasibility of the approach for real applications.

Keywords

Secret hiding, secret recovery, secret authentication, binary images, error-correcting schemes, distortion reduction, surrounding edge count, pixel embeddability.

1. INTRODUCTION

Information hiding behind digital images has many applications, including covert communication, copyright protection, annotation association, etc. However, it is generally difficult to hide information behind binary images. There are at least three reasons for this problem. First, embedding data in a binary image will cause obvious image content changes because of the binary (black and white) nature of the image. This indicates that reduction of image distortion due to data embedding (called embedding distortion in the sequel) should be taken as a major consideration in designing algorithms for data hiding in binary images. Next, binary images are more fragile to disturbances or attacks like channel noise or image operations. Such a characteristic makes authentication of recovered hidden information a required work. Finally, with the widespread use of color images, binary images are used today mainly for conveying text or graphic based document images in which color information is not important, and so the semantics of binary image contents are very vulnerable to pixel value changes due to data hiding. This means that a more careful selection of image pixels for data hiding is required; pixel value changes leading to obvious destruction of image contents should be avoided. In this paper, we propose an information hiding method which takes all of the above three requirements into consideration. Moreover, digital information that can be hidden by the method is general in type, and is assumed to be bit streams in the sequel.

There were only a few studies in the past about information hiding behind binary images, possibly due to the difficulty mentioned above. Wu, et al. [1] embedded bits in image blocks selected by pattern matching. The method can be used both for data hiding and for image authentication. Tseng, et al. [2] changed pixel values in image blocks and mapped block contents into the data to be hidden. In [3, 4], word or line spaces in textural document images are utilized to embed watermarks for copyright protection. In [5, 6], secret information is embedded into dithered images by manipulating dithering patterns. And Koch and Zhao [7] embedded a bit 0 or 1 in a block by enforcing the ratio of the number of black pixels in the block to that of white ones to be larger or smaller than the value 1, respectively. The method proposed in this study may be used for data hiding as well as data authentication. We will compare our method with [1] and [2] in more details later in this paper.

More specifically, in the proposed method we define a measure of pixel embeddability by which we can select suitable pixels from a given binary image, called the cover image, for embedding given secret data. The measure is defined in such a way that not only embedding distortion in the resulting image, called stego-image, can be reduced, but also the pixels selected for data embedding can be identified correctly subsequently for secret recovery. In addition, we employ an error-correcting scheme to encode the secret data before they are embedded, for the purposes of hidden data authentication as well as further embedding distortion reduction. At last, we propose the use of a secret key as well as a random number generator to randomize the locations of the selected pixels for data embedding. This enhances the security of the hidden data from being attacked or accessed illicitly. Based on these measures of distortion reduction and safety protection, processes for secret hiding and recovering are proposed. Some experimental results are also included to show the effectiveness the proposed method.

In the remainder of this paper, we first describe the proposed secret hiding and recovering processes in Section 2, followed by the descriptions of the involved measures for distortion reduction and security protection in Section 3. Some experimental results are given in Section 4, followed by a conclusion in Section 5.

2. PROPOSED SECRET EMBEDDING AND RECOVERING PROCESS

In the proposed method, we hide a given secret bit stream behind a binary image in a random fashion controlled by a secret key and a random number generator. The proposed secret hiding process is described first. Only basic ideas are included; the details of the involved terms and techniques will be explained in the next section. In the sequel, by embedding a value v into a pixel p, we mean to replace the value of p with v; and by extracting a value v from p, we mean to take v to be the value of p.

Algorithm 1. ***Secret hiding process***.

Input: a secret bit stream S, a cover image I, a secret key K, a random number generator g, and three pre-selected positive integer numbers m, n, and t.

Output: a stego-image I' in which S is embedded.

Steps:

1. Take sequentially m bits of S and encode them, using a t-error-correcting scheme, to form an n-bit substream s.
2. Create a set C of n-bit streams from s by changing at most t bits in s in all possible ways.
3. Select an *ordered sequence* E of n *embeddable pixels* in I randomly using g with K as the seed.
4. Select from C a substream s_{opt}, which causes minimum distortion, after being embedded into the pixels of E.
5. Embed the bits of s_{opt} sequentially into E.
6. Repeat Steps 1 through 5 until all bits in S are processed.

For convenience, in the sequel each pixel selected to be included in E in Step 3 above is said to *have been visited*. The proposed secret recovering process (including secret bit stream extraction and authentication) is described as follows.

Algorithm 2. ***Secret recovering process***.

Input: a stego-image I' presumably including a secret bit stream S; and the secret key K, the random number generator g, as well as the positive integer numbers m, n, and t all used in Algorithm 1.

Output: the secret bit stream S or a report of failure to recover the secret.

Steps:

1. Select an ordered sequence E of n embeddable pixels in I' using g with K as the seed.
2. Extract a bit b' from each pixel p in E, and compose all the n extracted bits sequentially to form a bit stream s'.
3. Decode s' by the t-error-correcting scheme used in Algorithm 1 to recover an m-bit secret stream s''. If more than t errors are found in s' during the decoding process, decide the bits of s'' to be *unauthentic*, yield a report of failure to recover the secret, and exit; otherwise, take s'' as part of the desired secret bit stream S.
4. Repeat the above steps to extract other m-bit substreams sequentially to compose the remaining part of S until done.

The ordered sequence E of pixels selected in Step 1 above presumably should be identical to that yielded in Step 3 of Algorithm 1 to ensure that the secret bit stream can be extracted correctly. For this to be true, in addition to requiring the use of the same random number generator g and the same secret key K in the two processes as already done, an extra condition is that the embeddability of the selected pixels must be *preserved* after the secret hiding process, and not be changed before the secret recovering process. We satisfy this condition by proposing a proper definition of pixel embeddability, as described in the next section.

3. PROPOSED PIXEL EMBEDDABILITY AND DISTORTION REDUCTION MEASURES

3.1 Pixel embeddability based on surround edge count

We define pixel embeddability from the viewpoint of reducing embedding distortion. First, we propose a new type of feature, called *surrounding edge count* and abbreviated as SEC. Let B be a 3×3 block in a cover image I with pixel p being its center and p_1, p_2, …, and p_8 being the eight surrounding neighbors of p in B. The SEC of p, denoted as SEC_p, is defined as the number of *existing edges* between p and its eight neighbors in B. Since I is binary, the existence of an edge between p with value v and one of its neighbors, say p_i with value v_i, means that $|v_i - v| = 1$, and the reverse situation means that $|v_i - v| = 0$. This in turn means that SEC_p may be computed by

$$SEC_p = \sum_{i=1}^{8} | v_i - v | \cdot$$

The SEC value of p is a measure of the structural randomness in the block from the centralized viewpoint of p. By definition, the SEC value of a fully black or white block is 0 (no edge exists), that of a block filled with a checker pattern is 4 (four edges exist), and that of a white (or black) pixel surrounded by eight black (or white) neighbors is 8 (eight edges exist).

Next, we define a *measure of distortion* resulting from complementing the value v of p by:

$$\Delta SEC_p = |SEC_p - SEC_p'|$$

where SEC_p and SEC_p' denote the SEC values before and after the complementation operation, respectively. It is not difficult to figure out that the above measure of distortion is just the amount of the resulting change of the numbers of edges in B. As an example, if the central pixel p of a fully black block is changed to be a white one, the above distortion value ΔSEC_p will have the largest possible value 8, which cannot be endured, because then the new white central pixel is too contrastive to its eight black neighbors.

Finally, we define a pixel p in a block B to be *embeddable* (i.e., suitable for embedding a bit value) if the following two conditions are satisfied:

(a) $\Delta SEC_p \leq T_{\text{d}}$; and

(b) p and its eight neighbors in B have not been visited yet,

where T_d is a pre-selected threshold value. Condition (a) above restricts the distortion introduced by the complementation of p's value to be *sufficiently small*, so that the resulting image quality will not be affected too much. And Condition (b) requires that embeddable pixels be *disconnected* from one another (by at least one pixel in distance), so that pixel value changes due to secret embedding will not be clustered or propagated to cause obvious larger-sized visual artifacts.

It is pointed out that pixel embeddability defined as above can be *preserved* indeed after the secret hiding process. The existence of this embeddability preserving property is briefly explained as follows. First, the pixel p and its eight neighbors are required by Condition (b) to be unvisited yet, and this means that the neighbors' values will not be altered after p is visited and labeled to be embeddable. This in turn means that the value ΔSEC_p will be fixed, and so Condition (a) will hold after the secret hiding process. As a result, after a secret bit is embedded into an embeddable pixel p in the secret hiding process, p will still be embeddable in the secret recovering process, guaranteeing that the embedded secret bit stream can be extracted correctly.

3.2 Authentication of extracted secret bit streams

In Step 3 of *Algorithm* 2, we recover secret substreams and authenticate them simultaneously using a t-error-correcting scheme described in [8]. The authentication capability of the scheme is explained here. In the encoding stage, the scheme appends several extra bits, forming a *redundant checking part*, to a bit sequence, called the *message part*. Each extra bit in the redundant checking part is a parity bit computed from a certain number of bits at certain specific positions in the message part. Then, in the decoding stage, if some bits in the two parts are changed, after the parity bits are computed, their values will be found to mismatch those in the redundant checking part, and errors can thus be detected. In this way, authentication of the message part, which, in our case here, is the secret data stream, can be achieved.

3.3 Further reduction of embedding distortion

By using the error-correcting scheme, errors in the extracted secret data not only can be detected, but also can be corrected. In this study, we adopt the BCH method [8] to achieve such an error-correction function in the scheme. And this error-correcting capability is utilized in Step 4 in Algorithm 1 to select a so-called optimal substream s_{opt} for further reduction of embedding distortion from a more global view. The details are described in the following.

After embedding an n-bit secret substream $s = b_1b_2\ldots b_n$ into n pixels $p_1, p_2, \ldots, p_n$ in a selected embeddable pixel sequence E, we compute a measure of the *total embedding distortion*, denoted by $D(s)$, as

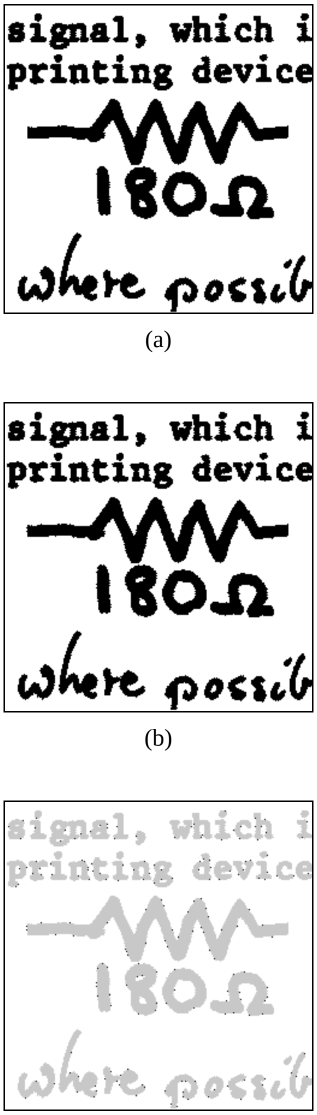
$$D(s) = \sum_{i=1}^{n} \Delta SEC_{p_i} \cdot$$

Also, as described in Step 2 in Algorithm 1, we create from s a set C of n-bit streams by changing at most t bits in s in all possible ways. For each stream s_i in C, we compute similarly another total embedding distortion value $D(s_i)$. Then, we choose as s_{opt} the stream s_i in C with the smallest $D(s_i)$, and embed s_{opt} into E as done in Step 3 of Algorithm 1. Thereafter, even though somehow erroneous bits occur in s_{opt} before secret recovering is conducted, if the number of errors is not larger than t, then by the t-error-correcting scheme, s can be still correctly recovered from s_{opt}, as done in Step 3 of Algorithm 2. Because of the freedom of the choice of s_{opt} from totally $\sum_{r=0}^{t}\binom{n}{r}$ candidate streams in C, it may be expected that the embedding distortion can be reduced further.

4. EXPERIMENTAL RESULTS

A large number of binary images, including several typical ones used for testing binary image compression standards, were used in our experiments. An example of the experimental results is shown in Figure 1. Figure 1(a) shows a 256×256 cover image with machine-printed as well as handwritten characters, and line drawings. And Figure 1(b) shows the stego-image resulting from embedding 630 secret bits into Figure 1(a) using Algorithm 1 with m, n, t, and T_d being 4, 15, 5, 3, respectively. The difference between Figure 1(a) and Figure 1(b) is illustrated in Figure 1(c) as black pixels. The gray pixels are included just for the purpose of comparison and are not part of the difference. It can be seen that Figure 1(a) and Figure 1(b) are visually close to each other; no obvious distortion can be observed. Another example of the results is shown in Figure 2, which demonstrates the effectiveness of the proposed technique for reducing embedding distortion. Figure 2(a) shows a 64×192 cover image. Two stego-images resulting from embedding a 60-bit secret stream into Figure 2(a) without and with the use of the t-error-correcting scheme are shown in Figure 2(b) and Figure 2(c), respectively. It can be noted that the visual quality of Figure 2(c) is better than that of Figure 2(b). Figure 3 shows the effect of distortion reduction using the error-correcting scheme with different values of t. The upper curve in the figure specifies the average ΔSEC yielded in Algorithm 1, and the lower curve specifies the *average number of changed pixels*, computed as the ratio of the number of changed pixels to that of the embedded secret bits. Both curves show that the distortion is decreased with the increase of the error-correcting capability specified by t.

Furthermore, we have implemented the methods proposed in [1] and [2] by programs in C++ to compare them with ours in the aspects of embedding capacity, robustness, and image quality. The comparison result is shown in Table 1. From the table, we can see that although the method of [2] has the maximum data embedding capability in the best case, it does not consider reduction of embedding distortions and might generate isolated spots in the stego-images. On the other hand, the method of [1] does consider distortion reduction just like ours, and if we compare image quality reduction by counting the total number of changed pixels in the embedding process, then our method is as good as [1]. However, our method can embed data more efficiently than [1] for most images by finding embeddable pixels in the image instead of embedding just a bit in every block. In short, our method maintains a good tradeoff between data embedding capacity and stego-image quality.



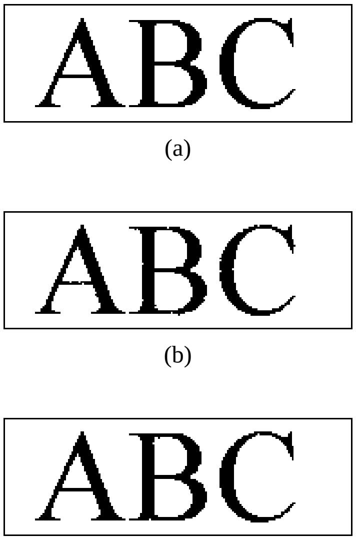
(a)

(b)

(c)

Figure 1. An experimental result of the proposed method: (a) a cover image; (b) the stego-image resulting from embedding 630 secret bits into (a); (c) the difference between the cover image (a) and the stego-image (b) shown as black pixels.

As to robustness, it was neither considered in [2] nor in our method because both methods were designed for steganography. Though, we still conducted some experiments to investigate the robustness of our method by introducing some random noise on the resulting stego-images, simulating possible attacks on the images. It is found that when the embeddability at certain pixels, which include an error-correcting code, is not destroyed by the noise, the proposed method will correct errors (no fewer than *t* ones) found in these pixels, thus achieving a certain degree of robustness. On the other hand, though [1] has mentioned how to achieve robustness against noise, no experimental data were shown. At last, it is emphasized that our method has the new capability of hidden data authentication, which is not found in any other method dealing with binary images, including [1] and [2].



(a)

(b)

(c)

Figure 2. Embedding results yielded with and without proposed embedding distortion reduction: (a) the cover image; (b) the stego-image yielded without distortion reduction; (c) the stego-image yielded with distortion reduction.

5. CONCLUSION

A new approach to data hiding in binary images has been proposed, which may be employed to hide general secret data behind binary images with the additional capability of hidden data

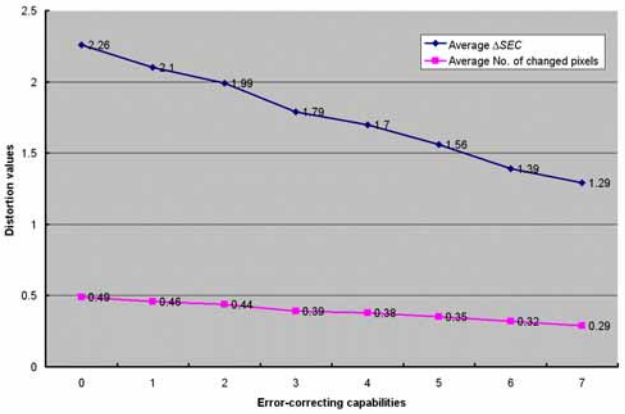


Figure 3. Curves showing the decrease of embedding distortion with the increase of the error-correcting capability specified by the number *t* of corrected bits.

authentication. Reduction of embedding distortion is the major consideration in the approach. The first measure for this goal is the proposal of pixel embeddability based on the new feature of SEC, which makes data hidden in embeddable pixels less noticeable. Computation of SEC values does not require excessive works like pattern matching, and so is efficient. Another measure proposed for distortion reduction from a more global view is the use of the error-correcting scheme for creating a distortion-minimizing secret stream from the original one. This merit is not found in other approaches dealing with data hiding in binary images. Our experimental results also reveal its effectiveness. In addition, the use of the error-correcting scheme also provides the ability to verify the authenticity of hidden data. Finally, because of the randomization policy employed for selecting embeddable pixels as well as the nature of the proposed pixel embeddability, embedded values are spread in the entire image and disconnected from one another, so that pixel changes will not be clustered and thus less hints for the embedded secret will be revealed.

6. REFERENCES

[1] M. Wu, E. Tang, and B. Liu, “Data hiding in digital binary images,” presented at the *IEEE International Conference on Multimedia and Expositions*, New York, 2000.

[2] Y.-C. Tseng, Y.-Y. Chen, and H.-K. Pan, “A secure data hiding scheme for binary images,” *IEEE Transactions on Communications*, vol. 50, no. 8, pp. 1227-1231, August 2002.

[3] S. H. Low, N. F. Maxemchuk, and A. M. Lapone, “Document identification for copyright protection using centroid detection,” *IEEE Transactions on Communications*, vol. 46, no. 3, pp. 372-383, March 1998.

[4] D. Huang and H. Yan, “Interword distance changes represented by sine waves for watermarking text images,” *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 11, no. 12, pp. 1237-1245, December 2001.

[5] K. Matsui and K. Tanaka, “Video-steganography: how to secretly embed a signature in a picture,” *Proceedings of IMA Intellectual Property Project*, vol. 1, no. 1, 1994.

[6] H. C. Wang, “Data hiding techniques for printed binary images,” *Proceedings of the IEEE International Conference on Information Technology: Coding and Computing*, Las Vegas, NV, USA, April 2001, pp. 55-59.

[7] E. Koch and J. Zhao, “Embedding robust labels into images for copyright protection,” *Proceedings of International Congress on Intellectual Property Rights for Specialized Information, Knowledge and New Techniques*, Munich, Germany, 1995, pp. 242-251.

[8] S. Lin and D. J. Costello, Jr., *Error Control Coding: Fundamentals and Applications*, Prentice-Hall, Inc., Englewood Cliffs, NJ, 1983.

Table 1. Comparison of characteristics of proposed method with those in [1] and [2].

	Wu and Liu’s method [1]	Tseng, Chen, and Pan’s method [2]	Proposed method
Processing manner	Block-based	Block-based	Pixel-based
Maximum embedding capacity of an $M \times N$ image (Block size: $m \times n$)	$(M/m) \times (N/n)$	$(M/m) \times (N/n) \times \lfloor \log_2 (mn+1) \rfloor$	$\lceil (M-2)/2 \rceil \times \lceil (N-2)/2 \rceil$
Reduction of embedding distortions	Yes	No	Yes
Generation of isolated spots in stego-images	No	Yes	No
Robustness to image manipulations	unknown	No	No
Authentication of hidden data	No	No	Yes