Reversible Data Hiding and Reversible Authentication Watermarking for Binary Images

Sergio Vicente Denser Pamboukian, and Hae Yong Kim, *Member, IEEE*

***Abstract*— Data hiding is a technique used to embed a sequence of bits in a host image with small visual deterioration and the means to extract it afterwards. Reversible data hiding allows, in addition, recovering the original cover-image exactly. Several reversible data hiding techniques have been developed but very few of them are appropriate for binary images. This paper proposes a reversible data hiding technique for binary images. This technique uses the Golomb code to compress prediction errors of low-visibility pixels, using its neighborhood as side information, to obtain the space to store the hidden data. The proposed technique is then used to reversibly authenticate binary images, including texts, drawings and halftones. All binary images we tested could be authenticated using the proposed technique, except unrealistically small or random images.**

***Index Terms*— Digital watermarking, reversible watermarking, binary images, steganography, data hiding, image authentication.**

I. Introduction

A DATA-HIDING scheme is a technique used to embed a sequence of bits in a host image and the means to extract it afterwards. Most data-hiding techniques modify and distort the host signal in order to insert the additional information. This distortion is usually small but irreversible. Reversible data-hiding techniques insert information bits by modifying the host signal, but enable the exact (lossless) restoration of the original host signal after extracting the embedded information.

In various fields, such as law enforcement, military imagery, medical imagery and astronomical research, a lossless recover of the host image is essential. Sometimes, expressions like distortion-free, invertible, reversible or erasable watermarking are used as synonyms for lossless watermarking. Some authors [1, 2] classify reversible data hiding techniques in two types:

- The first type [3, 4] makes use of additive spread spectrum techniques. In these techniques, a spread spectrum signal corresponding to the data to be embedded is superimposed (added) on the host signal. In the decoding, the hidden data is detected and the added signal is removed (subtracted) to restore the original host signal. These techniques use modulus arithmetic to avoid overflow/underflow errors, which may cause salt-and-pepper artifacts. They usually offer very limited information hiding capacity.
- In the second type [1, 2, 5, 6, 7], some portions of the host signal are overwritten by the embedded data. Two kinds of information must be embedded: the compressed data of the portion to be overwritten (to allow recovering the original signal) and the net payload data. During the decoding, the hidden information is extracted, the payload is recovered, and the compressed data is used to restore the original signal. These techniques do not cause salt-and-pepper artifacts, because the modified portions are usually the least significant bits or the high frequency wavelet coefficients that do not cause perceptible distortion. These techniques usually offer more data hiding capacity than the first type.

Among the reversible data hiding techniques, seemingly very few are adequate for binary images. We propose in this paper a reversible data hiding technique of the second type for binary images, called RDTC (Reversible Data hiding by Template ranking with symmetrical Central pixel). Then, we use RDTC to reversibly authenticate binary images and documents. There are two main challenges for designing a reversible data hiding of the second type for binary images:

- The first is developing a technique that must be able to localize precisely the changeable pixels in both insertion and extraction, in order to recover the original image. Some techniques do not have this property. Consider, for example, the data hiding where the cover image is subdivided into blocks, and one bit is inserted in each block by flipping (if necessary) the pixel with the lowest visibility (the flipping operation consists in transforming a white pixel in a black one or vice-versa). The blocks with even (odd) number of black pixels has bit zero (one) embedded. In this technique, the original image cannot be recovered even if the original parities of black pixels are known, because the precise flipped pixel inside each block cannot be localized. One solution is to find a suitable non-reversible data hiding technique that has this property and convert it into a reversible version. Another solution is to develop a totally new technique.
- The second is an efficient compression of the portion to be overwritten by the hidden data. This portion is typically small, has no structure and its samples are almost uniformly distributed and uncorrelated from sample to sample. Direct compression of the data therefore results in rather small lossless embedding capacity. However, if the remainder of the image is used as the

Manuscript received August 20, 2007; revised November 25, 2008. This work was supported in part by FAPESP under grant 2003/13752-9 and by CNPq under grants 305065/2003-3 and 475155/2004-1.

S.V.D. Pamboukian is with the Universidade Presbiteriana Mackenzie, Rua da Consolação, 930 – Prédio 6 – CEP 01302-907, São Paulo, SP, Brazil. E-mail: sergiop@mackenzie.br.

H.Y. Kim is with the Escola Politécnica, Universidade de São Paulo, Av. Prof. Luciano Gualberto, tr. 3, 158 – CEP 05508-900, São Paulo, SP, Brazil. E-mail: hae@lps.usp.br.

ISSN 1980-6604/$20.00 © 2008 SBrT/IEEE

side-information, significant compression gains can be achieved. In continuous-tone reversible data hiding, the choice of the compression algorithm seems not to be critical, because there is enough space to store the information (the least significant bits, for instance). Awrangjeb [1] uses Arithmetic Coding, LZW and JBIG for lossless compression. Celik [2] uses an adapted version of CALIC. In the reversible data hiding for binary images, on the contrary, most compression algorithms based on redundancy or dictionaries are not effective. RDTC uses the Golomb code to compress prediction errors of low-visibility pixels to obtain the space to store the hidden data.

II. PWLC Data Hiding Technique

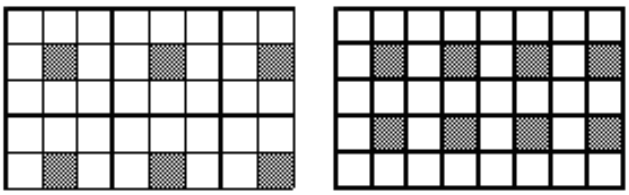
To the best of our knowledge, the only proposed reversible data hiding technique for binary images is PWLC (Pair-Wise Logical Computation) [8, 9]. It seems that sometimes PWLC does not extract correctly the hidden data, and fails to recover perfectly the original cover image.

PWLC uses neither the spread spectrum nor any compression techniques. It scans the host image in some order (for example, in raster scanning order). Only sequences "000000" or "111111" that are located near the image boundaries are chosen to hide data. The sequence "000000" becomes "001000" if bit 0 is inserted, and becomes "001100" if bit 1 is inserted. Similarly, the sequence "111111" becomes "110111" if bit 0 is inserted, and becomes "110011" if bit 1 is inserted.

However, the papers [8, 9] do not describe clearly how to identify the modified pixels in the extraction process. The image boundaries may change with the watermark insertion. Moreover, let us suppose that a sequence "001000" (located near to an image boundary) was found in the stego image. The papers do not describe how to discriminate between an unmarked "001000" sequence and an originally "000000" sequence that became "001000" with the insertion of the hidden bit 0.

III. DHTC Technique

The technique proposed in this paper (RDTC) is based on the non-reversible data hiding named DHTC (Data Hiding by Template ranking with symmetrical Central pixels) [10]. DHTC flips only low-visibility pixels to insert the hidden data and consequently images marked by DHTC have excellent visual quality and do not present salt-and-pepper noise.



(a) Non-overlapped image pieces (b) Partially overlapped image pieces

Fig. 1. Image divided in 3×3 pieces. Central pixels (candidates to bear information) are hatched.

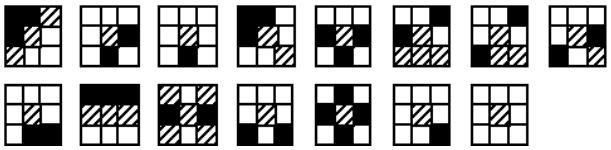


Fig. 2. A 3×3 template ranking with symmetrical central pixels in increasing visual impact order. Hatched pixels match either black or white pixels (note that all central pixels are hatched). The score of a given pattern is that of the matching template with the lowest impact. Mirrors, rotations and reverses of each pattern have the same score.

DHTC insertion algorithm is:

1) Divide the binary cover image Z in a sequence v of non-overlapping "image pieces" (e.g., 3×3) as we can see in figure 1a. Only the central pixels of the pieces of v can have their colors changed by the watermark insertion. A partial overlapping, that increases the storage capacity, can also be used (figure 1b).
2) Sort the sequence v in increasing order using the visual impact score as the primary-key and non-repeating pseudo-random numbers as the secondary-key. The primary key classifies the flippable central pixels according to their "visibility." Figure 2 enumerates all possible 3×3 templates, listed in increasing visibility of their central pixels. To assure the feasibility of reconstruction of v in the data extraction stage, two templates that differ only by the colors of their central pixels must have the same visibility score. This visibility ranking can be modified or larger templates may be used in order to minimize some specific perceptual distortion measure. The secondary-key prevents from embedding the data only in the upper part of the image.
3) The n first central pixels of the sorted v are the data bearing pixels (DBPs). Embed n bits of the data by flipping (if necessary) the DBPs.

To extract the hidden data, exactly the same sequence v must be reconstructed and sorted. Then, the n first central pixels are DBPs and their values are the hidden data.

In DHTC, the exact positions of n DBPs are known in both the data insertion and extraction. This property makes it possible to transform DHTC into a reversible data hiding. To do it, the original values of DBPs may be compressed, appended with the bits to be hidden (the net payload), and stored in the DBPs.

IV. The Proposed Techniques

This paper proposes a reversible data hiding technique for binary images called RDTC (Reversible Data hiding by Template ranking with symmetrical Central pixels) and a reversible fragile authentication watermarking called RATC (Reversible Authentication watermarking by Template ranking with symmetrical Central pixels).

A. The Proposed Reversible Data Hiding

The proposed reversible data hiding technique for binary images is based on DHTC technique previously described. In RDTC, two kinds of information must be embedded in the host

image: the compressed data to allow recovering the original image and the net payload data to be hidden. That is, the n DBPs' original values are compressed in order to create space to store the net payload data.

There are some difficulties to compress the DBPs' original values. Most compression algorithms based on redundancy and dictionaries do not work, because usually the amount of bits to be compressed is very small. Moreover, there is no way to predict the next bit based on the previous ones, because these bits correspond to the pixels dispersed throughout the whole image.

The solution we found is to compress the prediction errors of DBPs' values (using its neighborhood as the side-information) instead of their values directly. We tested two prediction schemes:

1) A pixel can be either of the same color or of the different color than the majority of its spatial neighboring pixels. Let us assume that the first hypothesis is more probable than the second. Let b be the number of black neighbor pixels of a DBP (using 3×3 templates, a DBP has 8 neighbor pixels). The prediction is correct (represented by 0) if the original DBP is black and $b > 4$, or if it is white and $b \leq 4$. Otherwise, the prediction is wrong (represented by 1). If this prediction is reasonable, the predicted value and the true value should be the same with probability higher than 50%. As we store zero when the prediction is correct and one when it is wrong, subsequences of zeros will be longer (in most cases) than subsequences of ones.
2) We also tested a more elaborate prediction scheme. We constructed a table with 256 elements (all possible configurations of 8 neighbor pixels) and, using typical binary images, determined the most probable central pixels' colors, based on the 8 neighbors' configurations.

Surprisingly, the two prediction schemes yielded almost the same results. The sequence of prediction errors consists of usually long segments of zeros separated by usually short segments of ones, because a zero occurs with high probability p and a one occurs with low probability $1 - p$. The Golomb code (to be explained in next section) is a good compression algorithm for this kind of sequence. As the DBPs' neighborhoods are not modified during the insertion, the prediction can be reconstructed in the extraction. The vector of prediction errors (0s and 1s), together with the neighborhoods of DBPs, allows recovering the original DBPs' values.

RDTC insertion algorithm is:

1) Divide the cover image Z in a sequence v of partially overlapping pieces.
2) Sort the sequence v in increasing order using the visual scores as the primary-key, the number of black pixels around the central pixels as the secondary-key and non-repeating pseudo-random numbers as the tertiary-key.
3) Estimate the smallest length n of DBPs capable of storing the header (size h), the compressed prediction errors vector (size w) and the given net payload data (size p), i.e., that satisfies $n \geq h + w + p$. Try iteratively different values of n, until obtaining the smallest n that satisfies the inequality above.
4) Insert the header (the values of n, w, p and the Golomb code parameter m), the compressed prediction errors vector and the payload by flipping the central pixels of the first n pieces of the sorted v.

To extract the payload and recover the original image, the sequence v of 3×3 image pieces is reconstructed and sorted. Then, the data is extracted from the n first central pixels of v. The compressed prediction errors vector is uncompressed and used to restore the original image.

We have embedded the data at the beginning of v, because this part has the least visible pixels. However, in order to obtain a higher embedding capacity (sacrificing the visual quality), we can scan the vector v searching for a segment that allows a better compression. The pixels at the beginning of v are the least visible ones but they cannot be predicted accurately, because usually they have similar number of black and white pixels in their neighborhoods (since they are boundary pixels). As we move forward in the vector, we find pixels that can be predicted more accurately, but with more visibility. In this case, the initial index of the embedded data in v must be stored in the beginning of v. Here, there is a trade-off between the visual quality of the stego image and the embedding capacity. Table 1 shows the number n of needed DBPs to hide 165 bits of payload (128 bits of net payload and 37 bits of header) at the beginning of v (the best quality) and in a segment that allows the best compression. In the latter case, the values of 176 pixels could be compressed to only $w = 11$ bits. In each image, n pixels were compressed, yielding w compressed pixels and $n - w$ free bits. Table 1 also shows the maximum amount σ of bits that can be reversibly inserted in each image. Note that in some cases, up to 20% of pixel images can be used to store information reversibly (300 dpi scanned text example in Table 1).

B. The Golomb Code

As we said in last sub-section, the sequence of prediction errors consists of long segments of zeros separated by short segments of ones. An efficient method to compress this type of information is the Golomb code [11, 12]. Some other methods based on the Golomb code (as LOCO-I, FELICS and JPEG-LS) also seem to be efficient, however we did not test them.

The Golomb code is used to encode sequences of zeros and ones, where a zero occurs with probability p and a one occurs with probability $1 - p$. This sequence is regarded as a nonnegative integer n. The Golomb code depends on the choice of an integer parameter $m \geq 2$ and it becomes the best prefix code when

$$m = \left\lceil -\frac{\log_2 (1+p)}{\log_2 p} \right\rceil$$

To compute the code of a nonnegative integer n, three quantities q, r and c are computed:

$$q = \left\lfloor \frac{n}{m} \right\rfloor, \quad r = n - qm, \quad \text{and} \quad c = \lceil \log_2 m \rceil$$

Then, the code is constructed in two parts: the first is the value of q, coded in unary, and the second is the binary value

TABLE I

REVERSIBLE INSERTION OF 165 BITS OF PAYLOAD IN DIFFERENT LETTER-SIZED IMAGES.

		Best quality			Best Compression			
Image description	Size	n	w	$n-w$	n	w	$n-w$	σ
Computer-generated text	1275×1650	336	146	190	176	11	165	401,600
150 dpi scanned text	1275×1650	432	265	167	176	11	165	214,032
300 dpi scanned text	2384×3194	496	325	171	176	11	165	1,543,680

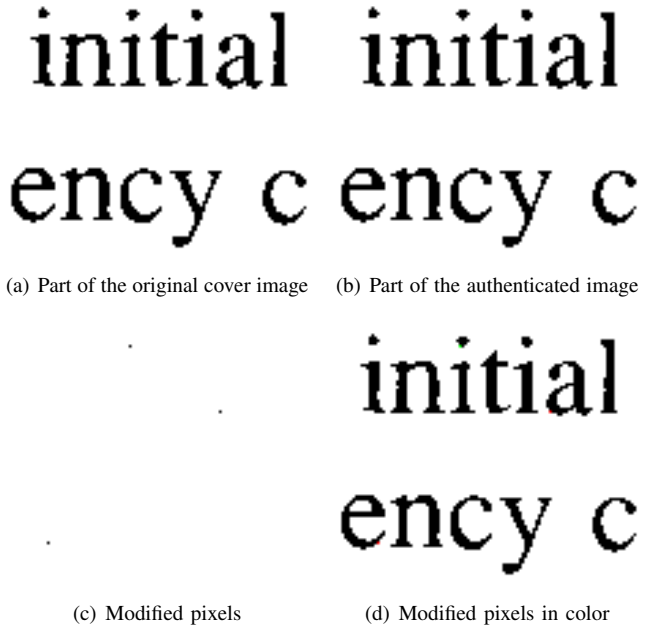
TABLE II

REVERSIBLE INSERTION OF 165 BITS OF PAYLOAD IN IMAGES OF DIFFERENT KINDS AND SIZES.

Image description	Size	n	w	$n-w$
Error diffusion halftone	512×512	400	233	167
Ordered dithering halftone	512×512	208	41	167
Computer-generated text	1275×1650	336	146	190
150 dpi scanned text	1275×1650	368	203	165
300 dpi scanned text	2384×3194	528	355	173
300 dpi scanned text	1092×1664	560	385	175
300 dpi scanned text	1094×414	464	288	176
400 dpi scanned text	3179×4259	464	290	174
Small computer-generated text	91×58	368	200	168
Tiny computer-generated text	64×56	368	193	175
10% random black pixels	300×300	400	227	173
20% random black pixels	300×300	880	711	169
30% random black pixels	300×300	3824	3654	170
40% random black pixels	300×300	Insertion was not possible		

of r coded in a special way. The first $2c - m$ values of r are coded as unsigned integers in $c - 1$ bits each and the rest are coded in c bits each. The case where m is a power of 2 is special because it requires no $(c - 1)$-bit codes (called Rice codes). To decode a Golomb code, the values of q and r are used to reconstruct n ($n = r + qm$). The readers are referred to [11, 12] to further details.

Fig. 5. Part of a 300×300 image with 40% random black pixels. It was not possible to reversibly hide 128 bits in this image.



(a) Part of the original cover image (b) Part of the authenticated image

(c) Modified pixels (d) Modified pixels in color

Fig. 3. Typical visual quality of a reversibly authenticated document. A letter-sized magazine page was scanned at 300 dpi, yielding an image with 2384×3194 pixels (a). It was reversibly authenticated with 128 hidden bits (b).

C. Reversible Authentication Watermarking

A reversible fragile authentication watermarking can be easily created using RDTC. Let us call it RATC (Reversible Authentication watermarking by Template ranking with symmetrical Central pixels). RATC can detect any image alteration. It can work with secret-key or public/private-key ciphers.

The public/private-key version of RATC insertion algorithm is:

1) Given a binary image Z to be authenticated, compute the integrity index of Z using a one-way hashing function $H = H(Z)$. Cryptograph the integrity index H using the private-key, obtaining the digital signature S.
2) Insert S into Z using RDTC, obtaining the watermarked stego image Z'.

RATC verification algorithm is:

1) Given a stego image Z', extract the authentication signature S and decrypted it using the public-key, obtaining the extracted integrity index E.
2) Extract the prediction errors vector, uncompress it and restore the original cover image Z. Recalculate the hashing function, obtaining the check integrity index $C = H(Z)$.
3) If the extracted integrity index E and the check integrity-

index C are the same, the image is authentic. Otherwise, the image was modified.

This technique can detect any image alteration, even a single pixel flipping, because all image pixels are used to compute the integrity index. Any alteration of data bearing pixels are detected because it changes the stored digital signature S used to compute the extracted integrity index E. And any alteration of pixels that do not bear data is also detected because it changes the check integrity index C. Indeed, the probability of an alteration passing undetected is only 2^{-n}, where n is the number of bits of the hashing function.

V. Experimental Results

We have tested RATC to authenticate binary images of different kinds and sizes (scanned texts, computer-generated texts, halftones, drawings, random noises, etc.) reversibly embedding 128 bits. 128 bits are enough to store a message authentication code, used in secret-key image authentication. Table 1 shows the maximum amount of bits σ that can be reversibly inserted in typical letter-sized document images. Table 2 shows that, in average, only 406 low-visibility pixels were compressed to get space enough to store 128 bits of payload data and 37 bits of header (excluding random noise images).

The marked stego-images have excellent visual quality, because only low-visibility pixels are modified. In a typical document authentication application, a letter-sized page is scanned at 300 dpi and authenticated. Figure 3 depicts the visual quality of an authenticated image in this situation. Note that only boundary low-visibility pixels were modified. Figure 4 depicts different kinds of small cover images, each one with 300×300 pixels, authenticated using RATC with 128 hidden bits, to show the visual quality of the authenticated images in unfavorable situations. In all cases, the recovered images are identical to the originals.

Only two kinds of images could not be authenticated, because it was not possible to embed 128 hidden bits: (1) very small images, like icons with 50×50 pixels, because there is not enough space to store 128 hidden bits and (2) random noise images with similar amounts of black and white pixels (figure 5), because the prediction is very difficult. On the other hand, theses kinds of image are very unusual and can be ignored for all practical purposes.

The executable RATC program is available at www.lps.usp.br/~hae/software/ratc.

VI. Conclusions

We have presented a reversible data hiding for binary images and used it to reversibly authenticate binary images. In this technique, prediction errors of low-visibility pixels are compressed using the Golomb code to create space to store the hidden data. We have applied the proposed technique to several kinds of binary images and, in average, only 406 pixels were compressed to get space to store 128 bits of net payload data. Resulting watermarked images have pleasant visual aspect.

References

[1] M. Awrangjeb and M. S. Kankanhalli, "Lossless Watermarking Considering the Human Visual System," *Int. Workshop on Digital Watermarking 2003*, Lecture Notes in Computer Science 2939, pp. 581-592, 2004.

[2] M. U. Celik, G. Sharma, A. M. Tekalp and E. Saber, "Reversible Data Hiding," in *Proc. IEEE Int. Conf. on Image Processing*, vol. 2, pp. 157-160, 2002.

[3] C. W. Honsinger, P. W. Jones, M. Rabbani, J. C. Stoffel, "Lossless Recovery of an Original Image Containing Embedded Data," US Patent #6,278,791, Aug 2001.

[4] J. Fridrich, M. Goljan, R. Du, "Invertible Authentication," in *Proc. SPIE Security and Watermarking of Multimedia Contents III*, (San Jose, California, USA), vol. 3971, pp. 197-208, 2001.

[5] J. Fridrich, M. Goljan, R. Du, "Lossless Data Embedding - New Paradigm in DigitalWatermarking," *EURASIP Journal on Applied Signal Processing,* no. 2, pp. 185-196, 2002.

[6] M. U. Celik, G. Sharma, A. M. Tekalp, E. Saber, "Lossless Generalized-LSB Data Embedding," *IEEE Trans. Image Processing,* vol. 14, no. 2, pp. 253-266, 2005.

[7] J. Tian, "Reversible Data Embedding Using Difference Expansion," *IEEE Trans. Circuits Systems and Video Technology,* vol. 13, no. 8, pp. 890-896, 2003.

[8] C-L Tsai, K-C Fan, C-D Chung and T.-C. Chuang, "Data Hiding of Binary Images Using Pair-wise Logical Computation Mechanism," in *Proc. IEEE International Conference on Multimedia and Expo, ICME 2004,* (Taipei, Taiwan), vol. 2, pp. 951-954, 2004.

[9] C-L Tsai, H-F Chiang, K-C Fan, C-D Chung, "Reversible Data Hiding and Lossless Reconstruction of Binary Images Using Pair-Wise Logical Computation Mechanism," *Pattern Recognition,* vol. 38, pp. 1993-2006, 2005.

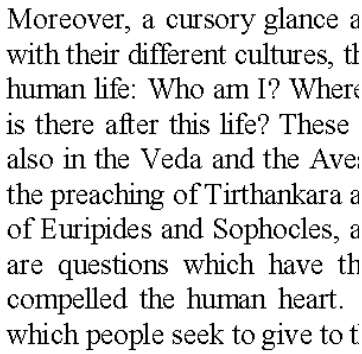
[10] H. Y. Kim, "A New Public-Key Authentication Watermarking for Binary Document Images Resistant to Parity Attacks," in *Proc. IEEE Int. Conf. on Image Processing,* vol. 2, pp. 1074-1077, 2005.

[11] S. W. Golomb, "Run-Length Encodings," *IEEE Trans. Inform. Theory,* vol. IT-12, pp. 399-401, 1966.

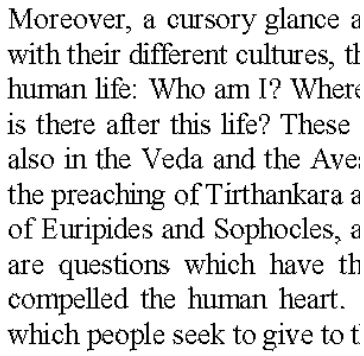
[12] D. Salomon, *Data Compression: The Complete Reference,* 3rd Edition, Springer-Verlag, New York, pp. 57-64, 2004.

Sergio Vicente Denser Pamboukian was born in Brazil in 1965. He received BSc in Civil Engineering (1988) and MSc in Electrical Engineering (1998) from the Universidade Presbiteriana Mackenzie (UPM), Brazil; and PhD in Electrical Engineering (2007) from the Universidade de São Paulo, Brazil. He is an assistant professor with the Dept. of Electrical Engineering at the UPM, since 1989. He is the author of many books about programming languages like C++ and Delphi. His research interests include authentication watermarking, image processing, software engineering, programming languages and techniques, object orientation, educational software and electronic data acquisition.

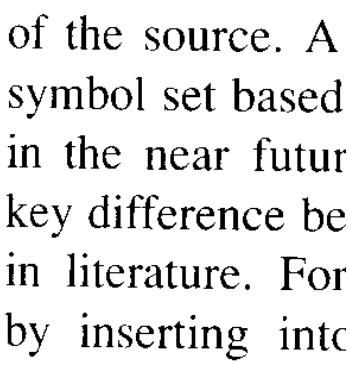
Hae Yong Kim was born in Korea in 1964 and migrated to Brazil in 1975. He received the best rating in the university ingressing examination for Computer Science, Universidade de São Paulo (USP), Brazil, and graduated with the best average marks in 1988. He received MSc in Applied Mathematics (1992) and PhD in Electrical Engineering (1997) from USP. Since 1989 he has been teaching in USP, and currently he is an associate professor with the Dept. of Electronic Systems Engineering, USP. Since 2002, CNPq (National Council for Scientific and Technological Development) has granted him a "research productivity award" scholarship. His research interests include the general area of image and video processing and analysis, authentication watermarking, and machine learning.



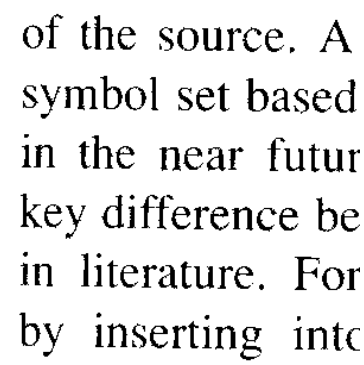
(a) Computer-generated document.



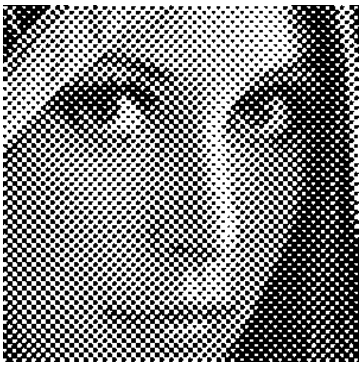
(b) Authenticated with $n = 464$, $w = 283$ and $n - w = 181$.



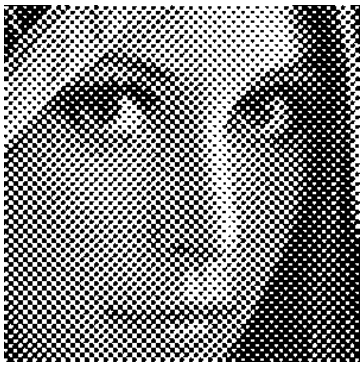
(c) A document scanned at 300 dpi.



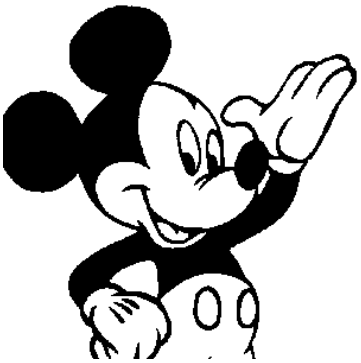
(d) Authenticated with $n = 432$, $w = 250$ and $n - w = 182$.



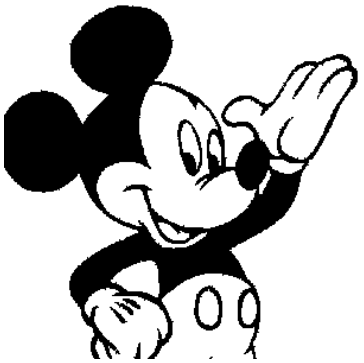
(e) Halftone image for laserjet printers.



(f) Authenticated with $n = 432$, $w = 254$ and $n - w = 178$.



(g) Mickey mouse.



(h) Authenticated with $n = 432$, $w = 256$ and $n - w = 176$.

Fig. 4. Different kinds of small 300×300 cover images (left column) and the corresponding reversibly authenticated images with 128 hidden bits and 37 header bits (right column). In each image, n pixels were compressed, yielding w compressed pixels and $n - w$ free bits where the authentication codes were inserted. In average, $n/2$ pixels are flipped.