REVERSIBLE DATA HIDING AND REVERSIBLE AUTHENTICATION WATERMARKING FOR BINARY IMAGES

Sergio Vicente D. Pamboukian[1] and Hae Yong Kim[2]
[1]*Universidade Presbiteriana Mackenzie, Brazil. E-mail: sergiop@mackenzie.com.br*
[2]*Universidade de São Paulo, Escola Politécnica, Brazil. E-mail: hae@lps.usp.br*

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 none of them seems to be appropriate for binary images. This paper proposes a reversible data hiding for binary images. The proposed technique selects a set of low-visibility pixels and uses the Golomb code to compress the predictions of these pixels. This compressed data and the net payload data are embedded into the image. Images watermarked by the proposed technique have excellent visual quality, because only low-visibility pixels are flipped. Then, we use the proposed data hiding to reversibly authenticate binary images and documents. This technique has many potential practical uses, including lossless authenticated FAX transmission and reversible content protection of binary document databases.

1. Introduction

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. Most data-hiding techniques modify and consequently distort the host signal in order to insert the additional information. This distortion is usually small but irreversible. Reversible data-hidings insert information bits by modifying the host signal, but enable the exact (lossless) restoration of the original host signal after extracting the embedded information. Sometimes, expressions like distortion-free, invertible, lossless or erasable watermarking are used as synonyms for reversible watermarking.

In most applications, the small distortion due to the data embedding is usually tolerable. However, the possibility of recovering the exact original image is a desirable property in many fields, like legal, medical and military imaging. Let us consider that sensitive documents (like bank checks) are scanned, protected with an authentication scheme based on a reversible data hiding, and sent through the Internet. In most cases, the watermarked documents will be sufficient to distinguish unambiguously the contents of the documents. However, if any uncertainty arises, the possibility of recovering the original unmarked document is very interesting.

To the best of our knowledge, none of the available reversible data hidings is adequate for watermarking binary images. We propose in this paper a reversible data hiding for binary images called RDTC (Reversible Data hiding by Template ranking with symmetrical Central pixels). Images watermarked by the proposed technique have excellent visual quality, because only low-visibility pixels are flipped. RDTC is adequate for watermarking most types of binary images, like scanned or computer-generated texts, charts and graphics; cartoon-like images; and clustered-dot halftones. RDTC can even be used to watermark dispersed-dot halftones (like images generated by error diffusion), however the resulting watermarked image may not present high visual quality, because the concept "low-visibility pixel" does not apply to this kind of image.

Then, we use RDTC to create a reversible public-key authentication watermarking for binary images named RATC (Reversible Authentication Watermarking by Template ranking with symmetrical Central pixels). Any reversible data hiding technique can be easily converted into a reversible authentication watermarking, provided that an enough number of bits can be embedded into the host image. To do it, the digital signature (DS) of the original image is computed using the private key. Then, the DS is embedded into the image, along with the information to allow recovering the original image. The verification algorithm extracts the DS, restores the original cover-

image and verifies whether the DS matches the recovered image.

The advantages of reversibly embedding the DS over appending it are obvious. First, there is no extra information (besides the image itself) to be stored or transmitted. Second, any lossless format conversion, such as changing the format from TGA to BMP, does not erase the embedded information. Third, the presence of a reversible authentication is less noticeable than the ostensibly appended DS.

RATC has many potential practical uses, because most of scanned documents are binary, and they must be digitally signed to assure their authenticity and integrity. Using RATC, the receiver of an Internet FAX document can be sure of the identity of the sender of the document and that the document was not tampered with. It is also possible to publish a database of binary documents in the Internet (for example, patent documents) and the reader can be sure that the documents are authentic and that they were not maliciously modified.

2. Reversible watermarking

Some authors [1, 2, 3] classify reversible data hiding techniques in two types: (1) those based on additive spread spectrum and (2) those based on image feature compression (high-capacity reversible watermarking).

(1) The first type [4, 5] 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. In this type, the payload extraction is robust, in the sense that the payload can be extracted even if the watermarked image is slightly modified. However, in this case, the original image cannot be recovered. These techniques use modulus arithmetic to avoid overflow/underflow errors, which may cause salt-and-pepper artifacts. Moreover, they usually offer very limited information hiding capacity.

(2) The second type [1, 2, 6, 7, 8, 9, 10] overwrites some portions of the host signal with 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 data 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.

The proposed technique RDTC is of the second type. There are two main challenges for designing a reversible data hiding of the second type for binary images:

(1) The first is to find a suitable non-reversible data hiding technique to be converted into a reversible version. In order to recover the original image, this technique must be able to localize precisely the flippable pixels in both insertion and extraction. Many 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 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.

(2) 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 virtually 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 side-information, significant compression gains can be achieved [2]. 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 (least significant bits, for instance). Awrangjeb [1] uses Arithmetic Coding, LZW and JBIG for lossless compression. Celik [2] uses an adapted version of CALIC [11]. In the reversible data hiding for binary images, in contrast, most compression algorithms based on redundancy or dictionaries are not effective. RDTC uses the Golomb code to compress predictions of low-visibility pixels to obtain the space to store the hidden data.

3. 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) [12, 13]. However, it seems that sometimes PWLC does not correctly extract the hidden data, and fails to recover perfectly the original cover image.

PWLC uses neither the spread spectrum nor any compression technique. It uses XOR binary operations to store the payload in the host image. It scans the host image in some order (for example, in raster scanning order). Only sequences "000000" or "111111" that are located near to 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 [12, 13] 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.

4. DHTC data hiding 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) [14]. 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.

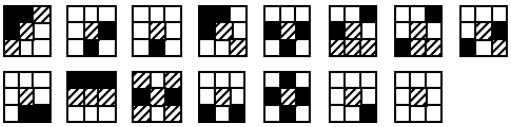


Figure 1: A 3×3 template ranking with symmetrical central pixels in increasing visual impact order. Hatched pixels match either black or white pixels (note that all central pixels are hatched). The score of a given pattern is that of the matching template with the lowest impact. Mirrors, rotations and 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). Only the central pixels of the pieces of v can have their colors changed by the watermark insertion.

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 1 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.

5. The proposed reversible data hiding

This paper proposes a reversible data hiding technique for binary images called RDTC (Reversible Data hiding by Template ranking with symmetrical Central pixels), based on DHTC 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, because these bits correspond to the pixels dispersed throughout the whole image.

The solution we found is to compress the predictions of DBPs' values (using its neighborhood as the side-information) instead of their values directly. 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 (a pixel is of the same color than the majority of its neighbors) is more probable than the second one. Let b be the num-

ber 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 the prediction is good, 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, what makes the compression possible. The Golomb code (to be explained in the next section) is a good compression algorithm for this kind of sequence. As the DBPs' neighborhoods are not modified during the insertion, the predictions can be reconstructed in the extraction. The vector of predictions (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 non-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 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 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 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, 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 also be stored in the header. Here, there is a trade-off between the visual quality of the stego image and the embedding capacity.

6. The Golomb Code

As we said in the last section, the sequence of predictions consists of (usually long) segments of zeros separated by (usually short) segments of ones. An efficient method to compress this type of information is the Golomb code [15, 16, 17]. Some other methods based on the Golomb code (as LOCO-I [18], FELICS [19] and JPEG-LS [20]) 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 (high) probability p and a one occurs with (low) probability $1-p$. The Golomb code depends on the choice of an integer parameter $m\geq 2$ and it becomes the best prefix code when

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

For small values of m, the Golomb codes start short and increase quickly in length. For large values of m, the Golomb codes start long, but their lengths increase slowly.

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 of r coded in a special way. If $r<2^c-m$, r is coded as unsigned integers in c-1 bits. If $r\geq 2^c-m$, r is represented as the unsigned integer $r+2^c-m$ in c bits. 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 ($n=r+qm$).

Example 1: Let us encode n=17 using m=14:

$$q = \left\lfloor \frac{17}{14} \right\rfloor = 1,\ r = 17 - 1\times 14 = 3, \text{ and } c = \lceil \log_2 14 \rceil = 4 .$$

1. Encoding q=1 in unary yields 10.

2. As $r \geq 2^c-m$, the remainder r=3 is coded as unsigned number $r+2^c-m$=5 using c=4 bits, that is, $(0101)_2$.

3. Therefore, n=17 is encoded as 100101, the result of the concatenation of 10 and 0101. □

Example 2: Let us encode the following sequence with 59 bits.
00000100110001010000001110100010000010001001000110100001001

This sequence has 19 runs of zeros:
5, 2, 0, 3, 1, 6, 0, 0, 1, 3, 5, 3, 2, 3, 0, 1, 4, 2 and 0.

The last zero indicates that the sequence terminates with a 1. As this sequence has 41 zeros and 18 ones, the probability of a zero is 41/(41+18) ≅ 0.7, yielding $m = \lceil -\log 1.7 / \log 0.7 \rceil = \lceil 1.487 \rceil = 2$.

Thus, encoding the sequence with m=2, we obtain a sequence of 19 codes:
1101|100|00|101|01|11100|00|00|01|101|1101|101|100|101|00|01|1100|100|00

The result is a 52-bit sequence that represents the original 59 bits. There is almost no compression because p is not large enough. □

Very small values of p, such as 0.1, result in a sequence with more ones than zeros. In such case, the Golomb code should compress runs of ones. For values of p around 0.5, the Golomb code is not a good choice and other methods should be considered.

7. 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 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.

8. Experimental results

We have tested RDTC to reversibly embed 128 bits in binary images of different kinds and sizes (scanned texts, computer-generated texts, cartoon-like images, halftones, random noises, etc). 128 bits are enough to store a message authentication code, used in secret-key image authentication. It was necessary to compress in average (excluding random noise images) only 453 low-visibility pixels to get space enough to store 128 bits of payload data and 37 bits of header (table 1). The marked stego-images have excellent visual quality (figures 2 and 3), because only low-visibility pixels are modified. The recovered images are identical to the originals.

Only two kinds of images could not be marked: (1) very small images, because there is not enough space to store the payload and the compressed information and (2) random noise images with similar amounts of black and white pixels, because the prediction is very difficult. On the other hand, theses kinds of image are very unusual and the proposed technique can be used for practically all images.

9. Conclusions

We have presented a reversible data hiding for binary images and used it to reversibly authenticate binary images. In this technique, predictions of low-visibility pixels are compressed using the Golomb code to create space to store the hidden data. The proposed technique was applied to several kinds of binary images and, in average, only 453 pixels were compressed to get space to store 128 bits of net payload data. Resulting watermarked images have pleasant visual aspect.

10. 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] Y. Q. Shi, "Reversible Data Hiding," Int. Workshop on Digital Watermarking 2004, (Seoul), Lecture Notes in Computer Science 3304, pp. 1-13, 2004.

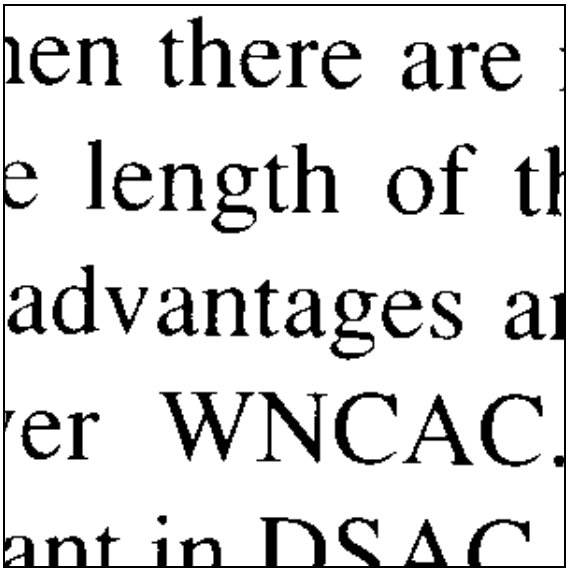
[4] C. W. Honsinger, P. W. Jones, M. Rabbani, J. C. Stoffel, "Lossless Recovery of an Original Image Containing Embedded Data," US Patent #6,278,791, Aug. 2001.

[5] 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.

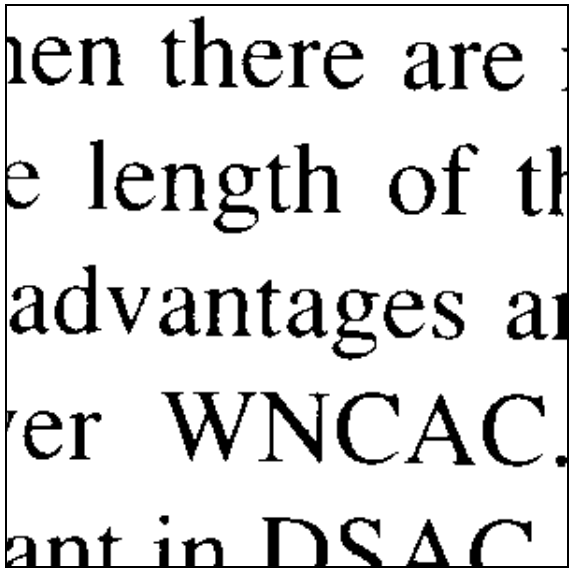
[6] J. Tian, "Reversible data embedding using difference expansion," IEEE Transactions on Circuits Systems and Video Technology, vol. 13, no. 8, pp. 890-896, 2003.
[7] J. Tian, "Wavelet-based reversible watermarking for authentication," in Proc. SPIE Security and Watermarking of Multimedia Contents IV, vol. 4675, pp. 679-690, 2002.
[8] J. Fridrich, M. Goljan, R. Du, "Lossless data embedding – new paradigm in digital watermarking," in EURASIP Journ. Appl. Sig. Proc., vol. 2002, no. 2, pp. 185-196, 2002.
[9] M. U. Celik; G. Sharma; A. M. Tekalp; E. Saber, "Lossless generalized-LSB data embedding," IEEE Transactions on Image Processing, vol. 14, no. 2, pp. 253-266, 2005.
[10] Z. C. Ni, Y. Q. Shi, N. Ansari, W. Su, Q. B. Sun, X. Lin, "Robust Lossless Image Data Hiding," IEEE Int. Conf. Multimedia and Expo 2004, pp. 2199-2202.
[11] X. Wu, "Lossless compression of continuous-tone images via context selection, quantization, and modeling," in IEEE Transactions on Image Processing, vol. 6, no. 5, pp. 656-664, 1997.
[12] 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.
[13] C. L. Tsai, K. C. Fan, C. D. Chung and T. C. Chuang, "Reversible and Lossless Data Hiding with Application in Digital Library," International Carnahan Conference on Security Technology, pp. 226-232, 2004.
[14] 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, (Italy), vol. 2, pp. 1074-1077, 2005.
[15] S. W. Golomb, "Run-Length Encodings," IEEE Trans. Inform. Theory, vol. IT-12, pp. 399-401, 1966.
[16] D. Salomon, Data Compression: The Complete Reference, 3rd Edition, Springer-Verlag, New York, 2004, pp. 57-64.
[17] R. Gallager; D. V. Voorhis, "Optimal source codes for geometrically distributed integer alphabets," IEEE Trans. Inform. Theory, vol. IT-21, pp. 228-230, 1975.
[18] M. J. Weinberger; G. Seroussi; G. Sapiro, "LOCO-I: A low complexity, context based, lossless image compression algorithm," in Proc. 1996 Data Compression conference, (Snowbird, Utah, USA), pp. 140-149, 1996.
[19] P. G. Howard; J. S. Vitter, "Fast and efficient lossless image compression," IEEE Data Compression Conference, pp. 351-360, 1993.
[20] ISO/IEC 14495-1, ITU Recommendation T.87, "Information Technology – Lossless and near-lossless compression of continuous-tones till images," 1999.

Table 1: Insertion of 128 bits of payload and 37 bits of header in different images, where *n* is the number of DBPs and *w* is the size of the compressed DBPs. "Not" means that the insertion was not possible.

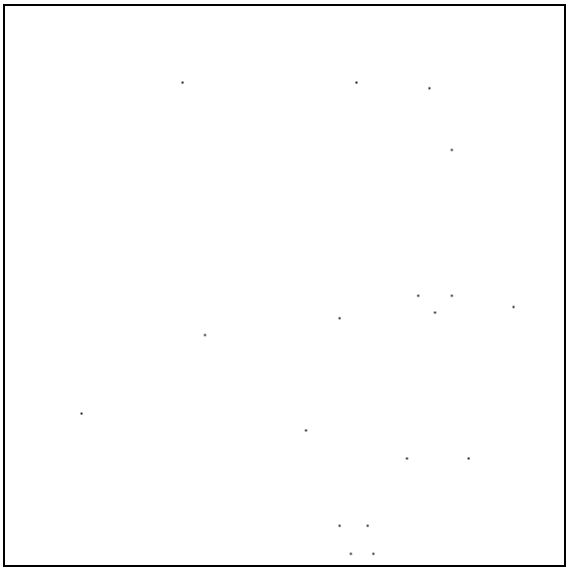
Image	**Description**	**Size**	***n***	***w***	***n-w***
lena0	Error diffusion	512×512	432	264	168
lena2	Ordered dithering	512×512	272	98	174
fides	Computer-generated text	1275×1650	432	259	173
persuas	150 dpi scanned text	1275×1650	560	395	165
toip300	300 dpi scanned text	2384×3194	496	325	171
toip300b	Sub-image of toip300	1094×414	464	288	176
toip300c	Sub-image of toip300	1092×1664	560	385	175
toip400	400 dpi scanned text	3179×4259	464	293	171
abc	Computer-generated text	91×58	400	219	181
pag1	Tiny computer-generated text	64×56	Not	Not	Not
noise10	10% random black pixels	300×300	400	227	173
noise20	20% random black pixels	300×300	880	711	169
noise30	30% random black pixels	300×300	3824	3654	170
noise35	35% random black pixels	300×300	Not	Not	Not
noise65	65% random black pixels	300×300	Not	Not	Not
noise70	70% random black pixels	300×300	1424	1256	168
noise80	80% random black pixels	300×300	592	423	169
noise90	90% random black pixels	300×300	368	194	174



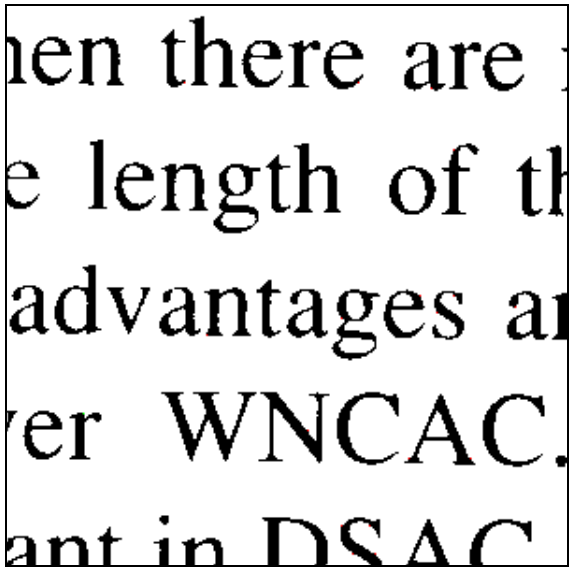
(a) Original cover image.



(b) Watermarked image.

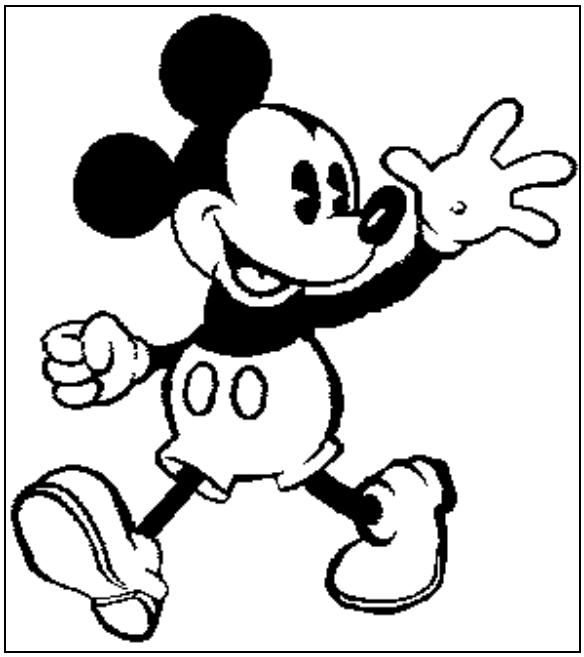


(c) Modified pixels.

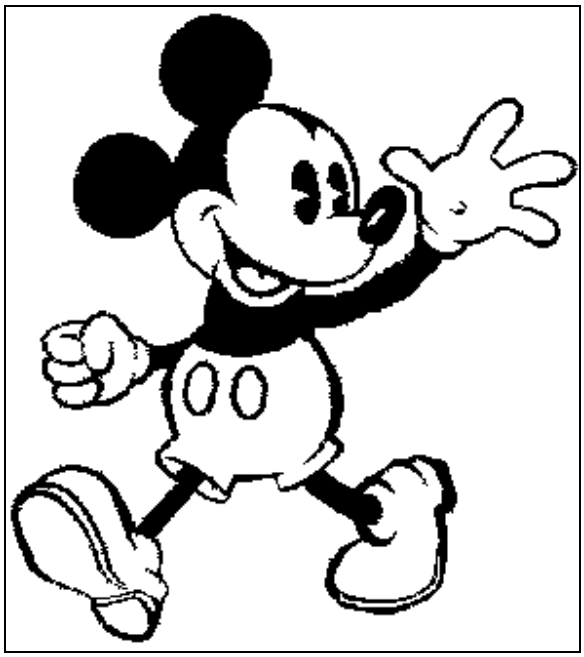


(d) Modified pixels in red.

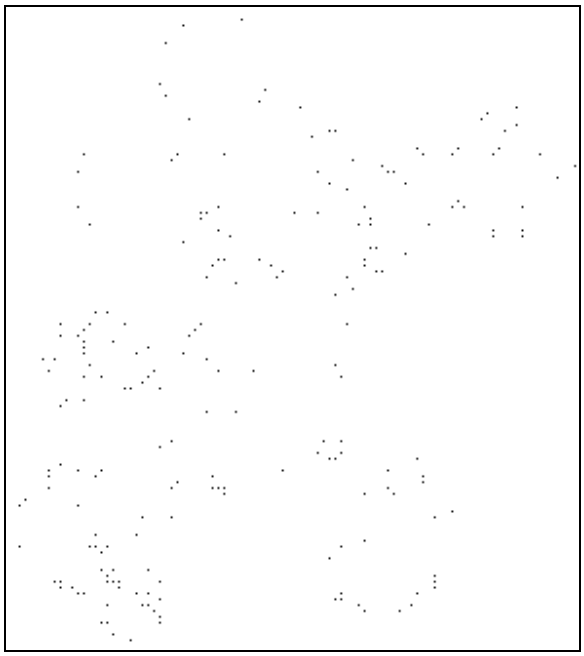
Figure 2: Part of a magazine page scanned at 400 dpi, with 3179×4259 pixels and reversibly watermarked with 1280-bits MAC (ten times the usually MAC size) using 3440 pixels to store the payload.



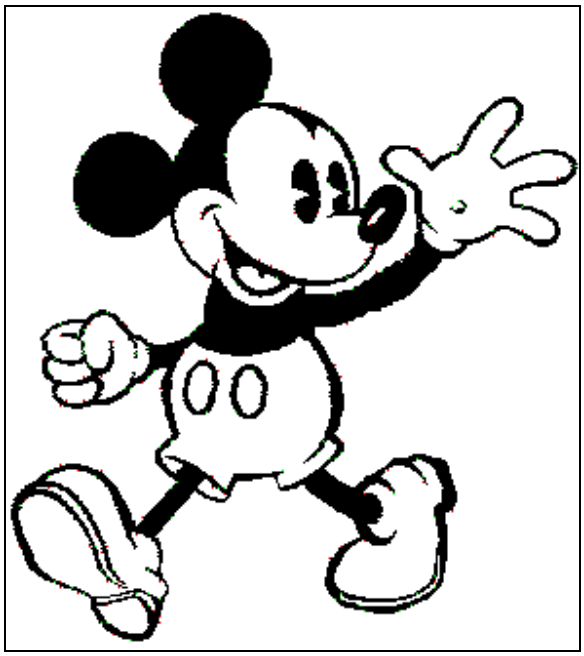
(a) Original cover image.



(b) Watermarked image.



(c) Modified pixels.



(d) Modified pixels in red.

Figure 3: A 295×331 image reversibly watermarked with 128-bits MAC, using 496 pixels to store the payload.