Lossless Data Embedding For All Image Formats

Jessica Fridrich*, Miroslav Goljan, Rui Du
Department of Electrical Engineering, SUNY Binghamton, Binghamton, NY 13902

ABSTRACT

Lossless data embedding has the property that the distortion due to embedding can be completely removed from the watermarked image without accessing any side channel. This can be a very important property whenever serious concerns over the image quality and artifacts visibility arise, such as for medical images, due to legal reasons, for military images or images used as evidence in court that may be viewed after enhancement and zooming. We formulate two general methodologies for lossless embedding that can be applied to images as well as any other digital objects, including video, audio, and other structures with redundancy. We use the general principles as guidelines for designing efficient, simple, and high-capacity lossless embedding methods for three most common image format paradigms – raw, uncompressed formats (BMP), lossy or transform formats (JPEG), and palette formats (GIF, PNG). We close the paper with examples of how the concept of lossless data embedding can be used as a powerful tool to achieve a variety of non-trivial tasks, including elegant lossless authentication using fragile watermarks. Note on terminology: some authors coined the terms erasable, removable, reversible, invertible, and distortion-free for the same concept.

Keywords: Lossless, distortion-free, removable, reversible, erasable, invertible embedding, steganography, watermarking, compression, authentication

1. INTRODUCTION

Data embedding applications could be divided into two groups depending on the relationship between the embedded message and the cover image. The first group is formed by steganographic applications in which the message has no relationship to the cover image and the cover image plays the role of a decoy to mask the very presence of communication. The content of the cover image has no value to the sender or the decoder. In this typical example of a steganographic application for covert communication, the receiver has no interest in the original cover image before the message was embedded. Thus, there is no need for lossless data embedding techniques for such applications.

The second group of applications is frequently addressed as digital watermarking. In a typical watermarking application, the message has a close relationship to the cover image. The message supplies additional information about the image, such as image caption, ancillary data about the image origin, author signature, image authentication code, etc. While the message increases the practical value of the image, the act of embedding inevitably introduces some amount of distortion. It is highly desirable that this distortion be as small as possible while meeting other requirements, such as minimal robustness and sufficient payload. Models of the human visual system are frequently used to make sure that the distortion due to embedding is imperceptible to the human eye. There are, however, some applications for which any distortion introduced to the image is not acceptable. A good example is medical imagery, where even small modifications are not allowed for obvious legal reasons. As another example, we mention law enforcement and military image analysts who may inspect imagery under special viewing conditions when typical assumptions about distortion visibility do not apply. Those conditions include extreme zoom, iterative filtering, and enhancement. Lossless data embedding could also be a convenient method of data embedding for customers who are overly concerned about decreasing the quality of their images by embedding a watermark.

Until recently, almost all data embedding techniques, especially high-capacity data embedding techniques, introduced some amount of distortion into the original image and the distortion was permanent and not reversible. Least Significant Bit (LSB) embedding and quantization schemes are examples of irreversible methods. In this paper, we present a solution to the problem of how to embed a large payload in digital images (covering all formats) in a lossless (reversible) manner so that after the payload bits are extracted, the image can be restored to its original form before

* fridrich@binghamton.edu; phone: 1 607 777-2577; fax: 1 607 777-2577; http://www.ssie.binghamton.edu/fridrich; SUNY Binghamton; T. J. Watson School of Engineering, Dept. of Electrical Engineering, Binghamton, NY USA 13902-6000

the embedding started. Even though the distortion is completely invertible, we pay close attention to minimizing the amount of distortion after embedding.

The concept of distortion-free data embedding appeared for the first time in an authentication method developed by Honsinger[6]. The data is embedded using a spatial additive non-adaptive robust watermark[7] using addition modulo 256. The modulo addition may introduce some distortion into the watermarked image *Iw* when pixels with grayscales close to zero are flipped to values close to 255 and vice versa. Thus this lossless data embedding scheme will work as long as the watermarking scheme is robust with respect to the flipped pixels, which generally form a correlated salt-and-pepper noise. If the number of flipped pixels is too large, such as for astronomical images, it may not be possible to extract the payload correctly from the watermarked image. Another problem with this scheme is that the flipped pixels are very visible and the distortion in the watermarked image, albeit erasable, may be objectionable in many applications. The problem with the visibility of the artifacts can be partially alleviated by using different modulo additions on disjoint intervals[1]. More detailed analysis and further generalization of this technique can be found in our previous work[1].

Macq[8] described a modification to the patchwork algorithm to achieve lossless watermark embedding. He also uses addition modulo 256 and essentially embeds a one-bit watermark. Both Honsinger's method[6] and the method by Macq[8] cannot be used for embedding large payloads. Even though the distortion they introduce is erasable, the visible artifacts they may introduce may not be acceptable in many applications. Finally, the methods are not easily extendable to other image formats, such as the JPEG.

In the next section, we describe two general mechanisms for building lossless embedding techniques for any digital object, including digital images, video, and audio files. Then in Section 3, we briefly describe a lossless method that we previously proposed and analyzed for raw, uncompressed formats, such as the BMP format. In Section 4, we continue with lossless data embedding methods for the JPEG format. Several new methods for the palette formats (GIF, PNG) are covered in Section 5. Finally, in Section 6, we briefly discus some important applications of lossless data embedding, including invertible fragile image authentication and accurate steganalysis of LSB embedding. The paper is concluded in Section 7.

2. GENERAL METHODOLOGY FOR LOSSLESS DATA EMBEDDING IN DIGITAL OBJECTS

Let us assume that X is a collection of samples obtained by digitizing an analog signal (e.g., image, video, audio file, SAR image, etc.) or a collection of samples obtained through measurements or computer simulations. The first methodology is based on lossless compression of subsets or features of the samples comprising the digital object X. If the object X contains a subset B, or if we can derive a set of features B from X with the following two properties, lossless data embedding is possible.

Property Ia: B can be losslessly compressed (i.e., B has a losslessly compressible structure),
Property Ib: B can be randomized while preserving the perceptual quality of object X.

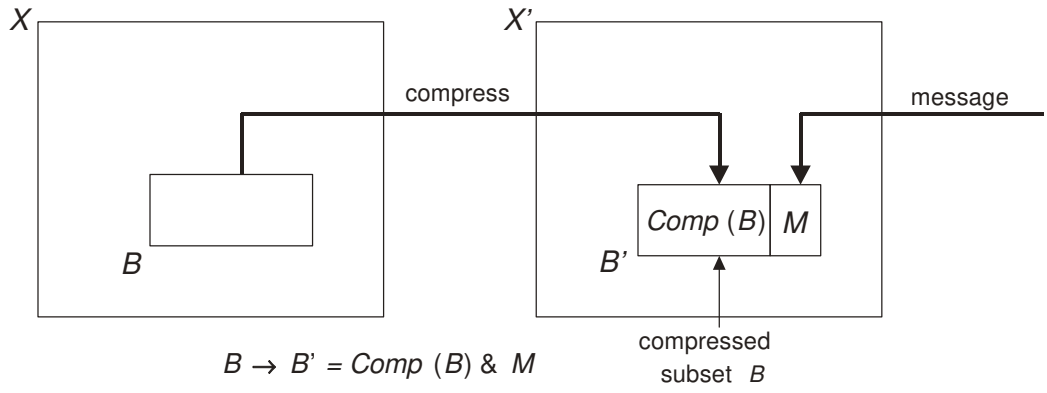


Figure 1 Diagram for lossless data embedding using lossless compression.

The embedding technique starts with extracting the subset B and continues with losslessly compressing it. The compressed bitstream is concatenated with a secret message (payload) and inserted into object X in place of subset B.

Property Ib above guarantees that object X will not be perceptibly disturbed, and Property Ia guarantees that the embedding method is lossless. The extraction of the hidden message proceeds by extracting the subset B and reading the concatenated bit stream consisting of the compressed bitstream and the message. The compressed bitstream is decompressed, and the original subset B is restored and reinserted into X. Thus the original object X is obtained. The methodology is summarized in Figure 1.

Our previous work on lossless authentication of raw[1] and JPEG images[4] provides a good example of this methodology. The object X is the set of all possible grayscale values from a grayscale image. The subset B is the set of all bits from a fixed bitplane. Lossless compression of the bitplane enables lossless data embedding as long as the bitplane is low enough so that replacing it with a compressed bitstream of itself and the message do not introduce visible artifacts. A different, significantly better technique for uncompressed formats is described in Section 3.

The second general approach to lossless data embedding is also based on the presence of subsets with specific properties in X. For each sample value x, we define the sample set $S(x) = \{t \in X \mid t = x\}$. For example, for a digital image, $S(123)$ is the set of all pixels with the grayscale value equal to 123. Let us assume that we can identify two sample values x and y with the following two properties:

Property IIa: The difference $|x-y|$ is small, i.e., the sample values are close to each other,
Property IIb: The sample sets $S(x)$ and $S(y)$ form a compressible bitstream (under some scan of X).

The embedding method starts with losslessly compressing the bitstream of samples while assigning, for example, a '0' to x and '1' to y and scanning object X in a defined pattern (we call this bitstream Z). The bitstream Z will be compressible, for example, when the cardinalities of $S(x)$ and $S(y)$ are different or when the samples are distributed in X in a non-uniform manner. Once we obtain the compressed bitstream, we concatenate it with a secret message (payload) and embed the result into the union of subsets $S(x)$ and $S(y)$ by scanning object X in the same pattern while considering x as a "0" and y as a "1". The act of embedding will not disturb the object X significantly because the difference between the values x and y is small according to Property IIa.

The message extraction proceeds by extracting the concatenated bitstream, reading the payload, and decompressing the compressed bitstream Z. Once the decompressed bit stream Z is obtained, we scan the object in the same defined pattern as we did during the embedding, and restore the original sample values x and y to their appropriate places in X.

This second approach can again be used for all image formats, but it is especially suitable for palette images.

3. THE RS LOSSLESS DATA EMBEDDING METHOD FOR UNCOMPRESSED IMAGE FORMATS

The technique for uncompressed formats has been introduced previously and is briefly repeated here for completeness. The reader is referred to the paper by Goljan et al.[3] for more details. Let us assume that the original image is a grayscale image with $M \times N$ pixels and with pixel values from the set P. For example, for an 8-bit grayscale image, $P = \{0, \ldots, 255\}$. We start with dividing the image into disjoint groups of n adjacent pixels $(x_1, \ldots, x_n)$. For example, we can choose groups of $n=4$ consecutive pixels in a row. We also define so called discrimination function f that assigns a real number $f(x_1, \ldots, x_n) = f(G) \in \mathbf{R}$ to each pixel group $G = (x_1, \ldots, x_n)$. The purpose of the discrimination function is to capture the smoothness or "regularity" of the group of pixels G. Image models or statistical assumptions about the original image can be used for design of other discrimination functions. In this paper, we chose the 'variation' of the group of pixels $(x_1, \ldots, x_n)$:

$$f(x_1, x_2, ..., x_n) = \sum_{i=1}^{n-1} |x_{i+1} - x_i| . \quad (1)$$

Finally, we define an invertible operation F on P called "flipping". Flipping is a permutation of gray levels that entirely consists of 2-cycles. Thus, F will have the property that F^2 = Identity or $F(F(x)) = x$ for all $x \in P$. For example, the permutation F_{LSB} defined as $0 \leftrightarrow 1$, $2 \leftrightarrow 3$, …, $254 \leftrightarrow 255$ corresponds to flipping (negating) the LSB of each gray level. The permutation $0 \leftrightarrow 2$, $1 \leftrightarrow 3$, $4 \leftrightarrow 6$, $5 \leftrightarrow 7$, … corresponds to an invertible noise with a larger "amplitude". One can easily visualize that many possible flipping permutations are possible, including those in which the flipping is irregular with several different changes in gray scales rather than just one. A useful numerical

characteristic for the permutation F is its "amplitude". The amplitude A of the flipping permutation F is defined as the average change of x under the application of F:

$$A = \frac{1}{|P|} \sum_{x \in P} |x - F(x)| \ . \tag{2}$$

For F_{LSB} the amplitude is 1. The other permutation from the previous paragraph has $A = 2$. Larger values of the amplitude A correspond to adding more noise after applying F.

We use the discrimination function f and the flipping operation F to define three types of pixel groups: R, S, and U

Regular groups:	$G \in R \Leftrightarrow f(F(G)) > f(G)$
Singular groups:	$G \in S \Leftrightarrow f(F(G)) < f(G)$
Unusable groups:	$G \in U \Leftrightarrow f(F(G)) = f(G)$.

In the expression $F(G)$, the flipping function F is applied to all (or selected) components of the vector $G=(x_1, \ldots, x_n)$. The noisier the group of pixels $G=(x_1, \ldots, x_n)$ is, the larger the value of the discrimination function becomes. The purpose of the flipping F is perturbing the pixel values in an invertible way by some small amount thus simulating the act of "invertible noise adding". In typical pictures, adding small amount of noise (i.e., flipping by a small amount) will lead to an increase in the discrimination function rather than decrease. Although this bias may be quite small, it will enable us to embed a large amount of information in an invertible manner.

Having explained the logic behind the definitions, we now outline the principle of the lossless high-capacity data embedding method. Let us denote the number of regular, singular, and unusable groups in the image as N_R, N_S, and N_U, respectively. We have $N_R+N_S+N_U = MN/n$. Because real images have spatial structures, we expect a bias between the number of regular groups and singular groups: $N_R > N_S$. As will be seen below, this bias will enable us to losslessly embed data. We further note that

if G is regular, $F(G)$ is singular,
if G is singular, $F(G)$ is regular, and
if G is unusable, $F(G)$ is unusable.

Thus, the R and S groups are flipped into each other under the flipping operation F, while the unusable groups U do not change their status. In a symbolic form, $F(R)=S$, $F(S)=R$, and $F(U)=U$.

We can now formulate the lossless data embedding method. By assigning a '1' to R and a '0' to S we embed one message bit in each R or S group. If the message bit and the group type do not match, we apply the flipping operation F to the group to obtain a match. We cannot use all R and S groups for the payload because we need to be able to revert to the exact original image after we extract the data at the receiving end. To solve this problem, we use the following idea. Before the embedding starts, we scan the image by groups and losslessly compress the status of the image – the bit-stream of R and S groups (the RS-vector) with the U groups simply skipped. We do not need to include the U groups, because they do not change in the process of message embedding and can be all unambiguously identified and skipped during embedding and extraction. We take the compressed RS-vector C, append the message bits to it, and embed the resulting bit-stream in the image using the process described above.

At the receiving end, the user simply extracts the bit-stream from all R and S groups (R='1', S='0') by scanning the image in the same order as during the embedding. The extracted bit-stream is separated into the message and the compressed RS-vector C. The bit-stream C is decompressed to reveal the original status of all R and S groups. The image is then processed and the status of all groups is adjusted as necessary by flipping the groups back to their original state. Thus, the exact copy of the original image is obtained.

The raw information capacity for this data embedding method is $N_R+N_S = MN/n - N_U$ bits. However, because we need to store the message *and* the compressed bit-stream C, the payload capacity Cap is $Cap = N_R+N_S - |C|$, where $|C|$ is the length of the bit-stream C. As the bias between R and S groups increases, the compressed bit-stream C becomes shorter and the capacity higher. An ideal lossless context-free compression scheme (the entropy coder[9]) would compress the RS-vector consisting of N_R+N_S bits using

$$- N_R \log\left(\frac{N_R}{N_R + N_S}\right) - N_S \log\left(\frac{N_S}{N_R + N_S}\right) \text{[bits]}.$$

As a result, we obtain

$$Cap = N_R + N_S + N_R \log\left(\frac{N_R}{N_R + N_S}\right) + N_S \log\left(\frac{N_S}{N_R + N_S}\right).$$

This estimate will be positive whenever there is a bias between the number of R and S groups, or when $N_R \neq N_S$. This bias is influenced by the size and shape of the group G, the discrimination function f, the amplitude of the flipping F, and the content of the original image. The bias increases with the group size n and the amplitude of the permutation F. Smoother and less noisy images lead to a larger bias than images that are highly textured or noisy.

Our goal is to choose such a combination of the group size n and its shape, the permutation F, and the discrimination function f, in order to maximize the capacity while keeping the distortion to the image as small as possible. The expression for the capacity *Cap* was experimentally verified on test images using the adaptive arithmetic coder[9] as the lossless compression. It was found that *Cap* matched the achieved bit-rate within 15–30 bits depending on the image size.

We have performed a number of experiments to see how the capacity and distortion change with different group sizes and shapes, discrimination functions f, and flipping operations F. It was a rather unexpected result that the highest capacity was obtained for relatively small groups ($n \approx 4$). Another surprising fact was that a quite reasonable capacity could be obtained from the flipping permutation F_{LSB} that influences only the LSBs. *And this was true for all images including those that did not show any structure in their LSB plane*.

Test image name ($M \times N$)	Capacity *Cap* for amplitude $a = 1, \ldots, 7$						
	1	2	3	4	5	6	7
LennaFace (128×128)	170	521	1045	1390	1865	1996	2342
Lenna (256×256)	1038	2916	5095	6027	7663	7783	8988
PalmTrees (400×268)	916	2274	4020	4621	5778	6643	7971
GoldenGate (400×268)	4325	8930	14001	14351	16865	16460	18341
Mountains (400×268)	1656	3790	6426	7575	9602	10432	12149
Desert (400×268)	7133	10935	17170	16959	19134	18568	20095
Mandrill (512×512)	186	702	1810	2905	4398	5664	7643
ElCapitan (592×800)	2500	12219	18898	26627	36774	42133	51430
NYC (1024×768)	6773	17766	30883	37516	48434	52553	61614
Girl (1024×1536)	25506	65577	109865	131994	166806	176587	204761
Capacity (bits per pixel)	0.019	0.041	0.069	0.078	0.097	0.10	0.12
Average PSNR (dB)	53.12	46.67	42.84	39.27	38.26	36.06	35.32

Table 1 Capacity *Cap* for ten grayscale test images as a function of the amplitude *a*.

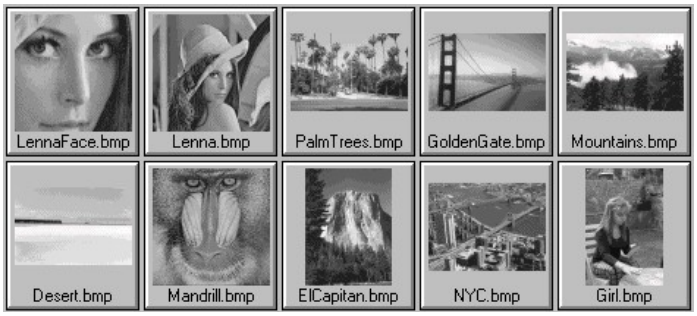


Figure 2 Test images used in Table 1.

In Table 1, we give an example of how the amplitude of the flipping function influences the capacity *Cap* and the distortion for ten grayscale images shown in Figure 2. We used groups of n=4 consecutive pixels and seven flipping operations with amplitudes ranging from 1 to 7. We observed a very high variability in the capacity between images. Images with abundant highly textured areas and noisy images generally have smaller capacity. It is also clearly apparent that the capacity increases very fast with amplitude.

The influence of noise and the flipping amplitude on the capacity and distortion was investigated in detail in our previous work[3]. The same paper contains more detailed analysis and further generalization of this RS lossless data embedding technique.

4. LOSSLESS DATA EMBEDDING FOR JPEG IMAGES

In this section, we describe three lossless data embedding techniques for JPEG images. The data is embedded in the quantized DCT coefficients in an invertible way so that it is possible to reconstruct from the watermarked JPEG file the exact copy of the original image data. In the first technique, the subset B with compressible structure is the set of all LSBs of one selected quantized DCT coefficient from all blocks. This set is obviously easily compressible (biased towards zero) but can be randomized without introducing disturbing distortion. The other two methods we present are based on a simple trick used to preprocess the original JPEG file to enable trivial lossless data embedding.

4.1. Method 1

Although in this paper, we explain the techniques on grayscale images, the technology can be extended to color images in a straightforward manner. The JPEG compression starts with dividing the image into disjoint blocks of 8×8 pixels. For each block, the discrete cosine transform (DCT) is calculated, producing 64 DCT coefficients. Let us denote the (i, j)-th DCT coefficient of the k-th block as $d_k(i, j)$, $0 \le i, j \le 64$, $k = 1, \ldots, b$, where b is the total number of blocks in the image. In each block, all 64 coefficients are further quantized to integers $D_k(i, j)$ with a JPEG quantization matrix Q

$$D_k(i, j) = integer_round\left(\frac{d_k(i, j)}{Q(i, j)}\right).$$

The quantized coefficients are arranged in a zig-zag manner and compressed using the Huffman coder. The resulting compressed stream together with a header forms the final JPEG file.

The largest DCT coefficients occur for the lowest frequencies (small i and j). Due to properties of typical images and due to quantization, the quantized DCT coefficients corresponding to higher frequencies have a large number of zeros or small integers, such as 1's or −1's. For example, for the classical grayscale test image 'Lenna' with 256×256 pixels, the DCT coefficient (5,5) is zero in 94.14% of all blocks. In 2.66% cases it is a 1, and in 2.81% cases it is equal to −1, with less than 1% of 2's and −2's. Thus, the sequence $D_k(5,5)$ forms a subset B that is easily compressible with the Huffman or arithmetic coder. Furthermore, if we embed message bits into the LSBs of the coefficients $D_k(5,5)$, we only need to compress the original LSBs of the sequence $D_k(5,5)$ instead of the coefficient values. We can further improve the efficiency of the algorithm if we define the LSB of negative integers $D_k < 0$ as $\mathrm{LSB}(D_k) = 1 - (|D_k| \bmod 2)$. Thus, LSB(−1)=LSB(−3)=0, and LSB(−2)=LSB(−4)=1, etc. Because DCT coefficients D_k have a symmetrical distribution with zero mean, this simple measure will increase the bias between zeros and ones in the LSB bit-stream of original DCT coefficients. With an increased bias, the lossless capacity will also be higher.

DCT coefficients $D_k(i,j)$ corresponding to higher-frequencies will produce a set B with a larger bias between zeros and ones, but because the quantization factor $Q(i,j)$ is higher for such coefficients, the distortion in each modified block will also be higher. To obtain the best results, one should use different DCT coefficients for different JPEG quality factors to minimize the overall distortion and avoid introducing visible artifacts. As a good overall choice, we recommend the coefficients corresponding to the middle frequencies. For color images, embedding into the chrominance channels introduces much less visible distortion than embedding into the luminance component.

Lossless data embedding in JPEG images (Method 1)

1. Based on the JPEG quality factor, determine the set of L pairs of indices (i_1,j_1), (i_2,j_2), …, (i_L,j_L), $0 \le i_l, j_l \le 64$, corresponding to middle frequencies.

2. Read the JPEG file and use the Huffman decompressor to obtain the values of quantized DCT coefficients, $D_k(i, j)$, $0 \le i, j \le 64$, $k = 1, \ldots, b$, where b is the total number of blocks in the image.
3. Seed a PRNG with a secret key and follow a random non-intersecting walk through the set $S=\{D_1(i_1,j_1), \ldots, D_b(i_1,j_1), D_1(i_2,j_2), \ldots, D_b(i_2,j_2), \ldots, D_1(i_L,j_L), \ldots, D_b(i_L,j_L)\}$. There are $L \times b$ elements in the set S.
4. While following the random walk, run the adaptive context-free lossless arithmetic compression algorithm for the LSBs of the coefficients from S (e.g., compress the set B of LSBs). While compressing, check for the difference between the length of the compressed bit-stream C and the number of processed coefficients. Once there is enough space to insert the message, stop running the compression algorithm. Denote the set of visited coefficients as S_1, $S_1 \subseteq S$.
5. Concatenate the compressed bit-stream C and the message and insert the resulting bit-stream into the LSBs of the coefficients from S_1. Huffman compress all DCT coefficients $D_k(i, j)$ including the modified ones and store the watermarked image as a JPEG file on a disk.

Message extraction and recovery of the original image (Method 1)

1. Based on the JPEG quality factor, determine the set of L pairs of indices $(i_1,j_1), (i_2,j_2), \ldots, (i_L,j_L)$, $0 \le i_l, j_l \le 64$.
2. Read the JPEG file and use Huffman decompressor to obtain the values of quantized DCT coefficients, $D_k(i, j)$, $0 \le i, j \le 64$, $k = 1, \ldots, b$.
3. Seed a PRNG with a secret key and follow a random non-intersecting walk through the set $S=\{D_1(i_1,j_1), \ldots, D_b(i_1,j_1), D_1(i_2,j_2), \ldots, D_b(i_2,j_2), \ldots, D_1(i_L,j_L), \ldots, D_b(i_L,j_L)\}$.
4. While following the random walk, run the context-free lossless arithmetic decompression algorithm for the LSBs of the coefficients visited during the random walk. Once the length of the decompressed bit-stream reaches b + *message_length* (the message length should be embedded in the header of the message), stop Step 4.
5. Separate the decompressed bit-stream into the LSBs of visited DCT coefficients and the message. Replace the LSBs of all visited coefficients with the decompressed bit-stream to obtain the original image.

Test image	Distortion (dB)			
	JPEG 90%	JPEG 85%	JPEG 75%	JPEG 50%
P1 (512×330)	55.0 (42.0)	52.0 (38.8)	48.0 (34.6)	43.0 (28.8)
P2 (256×256)	50.5 (38.6)	47.8 (35.3)	44.0 (30.0)	37.6 (25.2)
P3 (878×586)	59.7 (46.4)	56.6 (43.3)	53.6 (39.2)	47.2 (33.4)

Table 2 Distortion (PSNR) for lossless embedding of 128 (4000) bits for three test images P1, P2, P3, and different JPEG quality factors.

The selection of the L DCT coefficients can be adjusted according to the quality factor to minimize the distortion and other artifacts. For example, using L=3 coefficients (5,5), (4,6), and (6,3) in a random fashion will contribute to the overall security of the scheme because the statistical artifacts due to lossless authentication will be more difficult to detect. Table 2 shows the distortion measured using the PSNR. For simplicity, in our experiments we used one fixed DCT coefficient (6,6). The JPEG images were obtained by saving raw bitmaps as JPEGs with four different quality factors in PaintShop Pro 4.12.

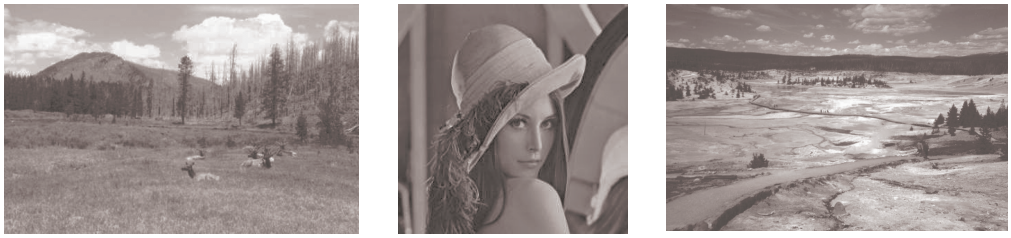


Figure 2 Three color test images P1, P2, and P3.

4.2. Method 2

The idea for the second method is quite simple. If, for a given DCT coefficient (i,j), the quantization factor $Q(i,j)$ is even, we could divide it by two and multiply all coefficients $D_k(i,j)$ by two without changing the visual appearance of the image at all. Because now all $D_k(i,j)$ are even, we can embed any binary message into the LSBs of $D_k(i,j)$ and this LSB embedding will be trivially invertible (the bias between zeros and ones is infinite).

If $Q(i,j)$ is odd, we replace it with $\lfloor (Q(i,j)/2) \rfloor$, where $\lfloor x \rfloor$ denotes the integer part of x, and multiply all $D_k(i,j)$ by two. In this case, we need to include a flag to the message telling us that $Q(i,j)$ was originally odd in order to be able to

reconstruct the original JPEG stream during message extraction. Because this method uses a non-standard quantization table, the table must be included in the header of the authenticated image. Because the table entry $Q(i,j)$ will not be compatible with the rest of the table, it will be easy to tell whether or not a given JPEG image has been authenticated by this method, i.e., the method is steganographically obvious.

One can imagine several other possible implementations of the above idea. For example, we could replace $Q(i,j)$ with a 1 instead of its half and multiply each $D_k(i,j)$ with $Q(i,j)$. The original value of the quantization step $Q(i,j)$ needs to be embedded as well to be able to reconstruct the original image. The message extraction will proceed simply by checking whether or not the DCT coefficient is a multiple of $Q(i,j)$. This version of the method will introduce very small distortion because the DCT coefficients used for embedding have a quantization factor equal to 1. On the other hand, the modified stream of quantized coefficients will be less compressible using the Huffman code thus worsening the overall compression ratio.

In Table 3, we give the PSNR for a 128-bit long message, three different images (P1, P2, and P3), four different quality factors (25, 50, 75, and 98), and three choices for the coefficient $D_k(i,j)$ − (6,6), (5,4), and (4,2), and three color images. The numbers in parenthesis correspond to a 4000 bit message.

		Method 2a $D_k(i,j) \rightarrow D_k(i,j)/2$			Method 2b $D_k(i,j) \rightarrow 1$		
Image	QF	(6,6)	(5,4)	(4,2)	(6,6)	(5,4)	(4,2)
	50	49.3 (34.7)	46.5 (32.3)	49.2 (33.2)	66.6 (54.4)	66.2 (54.6)	66.6 (54.2)
P1	75	54.6 (40.4)	52.0 (38.0)	52.9 (39.1)	66.9 (55.0)	66.5 (55.2)	67.0 (54.8)
512×330	85	58.0 (44.4)	55.6 (42.1)	57.0 (43.1)	66.5 (55.0)	66.0 (55.0)	66.6 (54.8)
	90	60.4 (47.1)	58.6 (45.1)	58.7 (46.3)	66.9 (55.1)	66.7 (55.1)	67.0 (54.7)
	50	43.7 (31.1)	40.8 (28.2)	43.2 (27.1)	62.0 (53.5)	61.8 (53.1)	62.0 (52.7)
P2	75	50.8 (37.1)	48.1 (34.3)	49.0 (32.9)	62.7 (53.5)	62.2 (53.0)	62.5 (52.5)
256×256	85	54.1 (41.2)	51.5 (38.5)	53.1 (37.2)	62.3 (53.5)	61.8 (53.0)	62.5 (52.6)
	90	56.0 (44.3)	54.0 (41.8)	54.7 (40.7)	62.5 (53.5)	62.3 (53.1)	62.2 (52.5)
	50	53.6 (39.4)	50.7 (36.4)	53.3 (38.4)	70.8 (56.8)	70.3 (56.4)	70.6 (56.7)
P3	75	59.2 (44.8)	56.6 (42.0)	57.6 (44.2)	71.4 (57.8)	71.4 (57.4)	71.8 (57.7)
878×586	85	62.7 (48.4)	60.3 (45.8)	61.7 (47.5)	71.4 (58.0)	70.6 (57.6)	71.2 (57.9)
	90	64.9 (50.5)	63.0 (48.5)	61.7 (50.4)	71.7 (58.1)	71.5 (57.7)	71.6 (57.9)

Table 3 PSNR for embedding a 128-bit message and for a 4000-bit message (in parenthesis) for three test images, four quality factors, and three different DCT coefficients.

The influence of lossless data embedding on the JPEG file size has been studied in our previous work[4].

5. LOSSLESS DATA EMBEDDING FOR PALETTE IMAGES (GIF)

5.1. Palettes with fewer than 256 unique colors

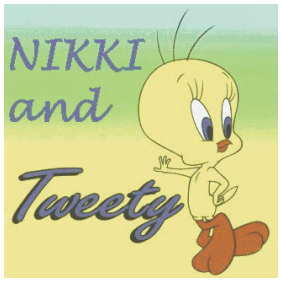
The palette format is typically used for images with a low number of colors. Thus, it frequently happens that the palette has fewer than 256 unique colors or that some colors do not occur in the image. Let us assume that a particular color c occurs in several places in the palette, or that the color c does not occur in the image at all. Let d denotes the most frequently occurring color in the image (this color can be easily found from the histogram). We propose to replace one occurrence of the color c in the palette (if it occurs multiple times in the palette) or the color c itself (if it does not occur in the image at all) and replace this palette entry with the color d. This way, the color d will have two occurrences in the palette. Let us denote the indices of those two occurrences by i and j. We can associate a '0' with i and a '1' with j and embed $f(d)$ bits losslessly in the image, where $f(d)$ is the number of occurrences of color d in the image.

In the embedding process, we scan the image by rows and whenever we encounter a pixel with color d, we change the pointer to the palette to i if we need to embed a "0" or to j if we need to embed a "1". Note that this embedding will not modify the *visual appearance* of the image at all, so it is trivially lossless. The image file will, however, be modified. It is also likely that the GIF file will likely increase in size after embedding because we are basically randomizing the pointers to the palette, which are losslessly compressed using the LZW algorithm in the GIF format.

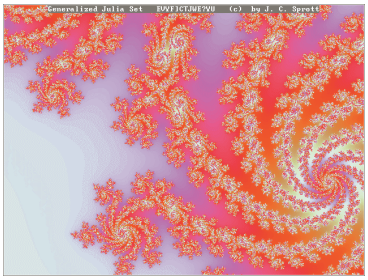
The message extraction proceeds by scanning the image by rows again and extracting the binary message depending on the pointers to the most frequently occurring color i and j. Because the visual appearance of the image is not

modified, there is no need to reconstruct the original image after message extraction because the image with embedded message is the same as the original. It may be beneficial, however, to remove the additional occurrence of color d from the stego-image and re-index the image to improve its compressibility as a GIF.

The capacity of this method depends on the cover image and is at least $MN/256$ bits where M and N are image dimensions. In practice, the capacity is much higher (see Table 4) because the distribution of colors (the histogram) is frequently highly non-uniform. We note that in the case that there is another color that occurs more than twice in the palette, it is possible to increase the capacity of the embedding scheme simply by using a ternary representation or higher-order representation for the message.



3018 bytes, 325×325, 246 colors


1688 bytes 640×489, 195 colors

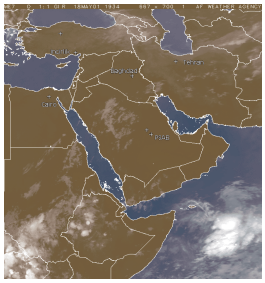

4043 bytes 667×713, 156 colors

Table 4 Capacity in bytes, image dimensions, and number of unique colors for 3 test images with non-full palettes.

5.2. Full palettes with 256 unique colors

If the palette is full and contains all 256 different colors, we cannot apply the technique described in the previous subsection and we need to choose a different approach.

First of all, we note that it is possible to apply the RS lossless data embedding method of Section 2 to palette images as well. We need to define a pairing of palette colors that will be flipped into each other. This will be our flipping operation. The distance between paired colors determines the distortion and influences the capacity. We can also include an upper bound on the distance between the pair of colors so that when $| c_i - c_j | > T$, where T is a fixed threshold, instead of flipping the colors, we leave them unchanged (i.e., we flip c_i into c_i and c_j into c_j) thus avoiding disturbing artifacts at the expense of slightly decreasing the capacity.

There is a whole class of algorithms depending on how we pair the palette colors. For example, we could choose the closest pair of colors first, remove them, then choose the next closest pair, till we pair all colors (bottom-up). We could also approach the pairing in the opposite direction by identifying the color with the largest isolation (distance to the closest color) and pair it with its closest color. After removing these two colors, we repeat the process till we pair all palette colors (top-down). According to our experience, both approaches typically provide similar capacities and acceptable distortions.

Instead of studying the RS method for GIFs, we developed another, different technique (color-pair method) that is simpler and better corresponds to the specifics of the GIF format. The method utilizes the second approach described in Section 2. For each pointer i to the palette, we define the sample set $S(i)$ as the subset of pixels from the image whose pointers are equal to i. The embedding method starts with identifying two pointers i and j whose associated palette colors c_i and c_j are close to each other (i.e., $|c_i–c_j|$ is small) but whose sample sets differ in size as much as possible ($|S(i)| – |S(j)|$ is large in absolute value). The symbol $|S(i)|$ means the number of elements in the subset $S(i)$.

The next step in the lossless data embedding algorithm is lossless compression of the bit-stream Z of elements $S(i)$ and $S(j)$, while assigning for example a ‘0’ to i and ‘1’ to j and scanning the image in a defined pattern. For example, we can scan it by rows and losslessly compress the bit-stream Z of values i (stands for a ‘0’) and j (stands for a ‘1’) as the image is scanned. Once this compressed bit-stream is obtained, we concatenate it with the secret message (payload and the hash) and embed it into the union of subsets $S(i)$ and $S(j)$ by scanning the image in the same pattern and choosing the pointer i if a ‘0’ should be embedded and the pointer j if a ‘1’ needs to be embedded. The embedding process will not disturb the image significantly because the difference between the colors c_i and c_j is small. Also, at the same time, because the subsets $S(i)$ and $S(j)$ differ in size, the bit-stream Z will be losslessly compressible (for example using the adaptive arithmetic coder[9]), which will enable us to embed the payload losslessly. The larger the

difference $|c_i - c_j|$ is, the larger the distortion due to embedding. The more compressible the bit-stream Z is, the larger the payload will be. We also point out that naturally occurring structures in the image may enable lossless embedding even when the difference in size between $S(i)$ and $S(j)$ is small. In this case, the LZW compression algorithm should be used instead of the arithmetic coder. In general, the embedding method works whenever the bit-stream Z is losslessly compressible.

The message extraction proceeds by scanning the image in the same manner as during the embedding, extracting the concatenated bit-stream from all pixels with pointers i and j, reading the payload, and decompressing the compressed bit-stream Z. Once the decompressed bit-stream Z is obtained, we scan the image in the same defined pattern as we did during the embedding and restore the original colors c_i and c_j to their appropriate places in the image.

Note that we need to somehow communicate the fact which pointers (colors) were used for embedding so that the message extraction can be done in an oblivious manner. This can be achieved by rearranging the palette so that the colors c_i and c_j occur as the first two colors in the palette. Also, it is possible to increase the capacity of this scheme by using more than one pair of colors.

Image	**Color pair**	T	**RS Top down**			**RS Bottom up**		
Castle	410	5	4110	1702	1293	3319	929	679
		10	4398	1700	1308	3434	977	729
		25	4419	1705	1320	3487	997	743
Butterfly	354	5	4540	1825	1398	3820	1014	793
		10	4807	1938	1523	4176	1225	980
		25	4890	1993	1550	4399	1349	1088
Cottage	10865	5	3422	658	531	1271	478	541
		10	3799	747	644	1461	581	602
		25	3766	748	659	1544	628	646
Nature	455	5	1490	924	817	1445	749	526
		10	1723	1018	897	1568	817	571
		25	1699	1124	1000	1580	814	568
Teahouse	771	5	3790	1623	1264	3183	1005	750
		10	4479	1748	1366	3844	1207	939
		25	4542	1761	1407	4129	1325	1071
Cat	469	5	1488	870	637	1155	594	435
		10	1541	972	755	1264	588	437
		25	1901	1357	1091	1295	622	468

Table 5 Capacity in bytes for six test images with full palettes for the color-pair method and for the RS embedding. The images (not shown due to space limitations) were obtained by converting 640×480 true-color JPEG images to the GIF format.

The second column in Table 5 shows the capacity in bytes for the color-pair method. The color pair was determined as the one that maximizes the capacity per distortion (measured as MSE). For completeness, we also show the results for the RS embedding with three threholds (T = 5, 10, 25) both for the bottom-up and top-down approaches and three embedding masks (black pixels are flipped):

diagonal mask with discrimination function $|x_{11} - x_{12}| + |x_{12} - x_{22}| + |x_{22} - x_{21}| + |x_{21} - x_{11}|$,

square 2×2 mask with the same discrimination function,

linear 1×2 mask with the discrimination function $|x_{11} - x_{12}|$.

The seemingly low capacity of the color-pair method can be substantially increased by using more than one pair of colors. Even with just one pair, the capacity was high enough for embedding authentication information (16–20 bytes) and a caption of several hundred bytes. The advantage of the color-pair method becomes obvious when we look at the low embedding distortion per message bit (see Table 6).

Image	Color pair	Top down			Bottom up		
		5	10	25	5	10	25
Castle	913.9	662.9	887.5	951.1	611.0	795.3	874.0
Butterfly	334.8	1027.3	1454.0	2099.4	860.5	1133.9	1716.6
Cottage	26.1	822.4	1052.2	1281.6	1762.1	2424.4	3093.8
Nature	128.1	2251.5	2358.5	2697.8	1749.8	2315.7	2441.9
Teahouse	513.9	1014.1	1917.5	2363.3	1043.3	1430.6	2082.0
Cat	288.5	2023.2	2172.8	2853.6	1758.1	1992.8	2282.2

Table 6 Square error per message bit for six test images for the color-pair method and RS embedding with three thresholds and three embedding masks.

6. APPLICATIONS

The ability to embed data in an image in a lossless manner without having to expand the image or append the data can be quite useful. Data embedded in a header or a separate file can be easily lost during file format conversion or resaving. Additional information embedded directly in the image as additional lines or columns may cause visually disturbing artifacts and increases the image file size. In contrast, information that is embedded in the image is not modified by compatible format conversion or resaving, and a better security is obtained because the embedded information is inconspicuous and imperceptible. For increased security, a secret key can protect the embedding process. In addition to these advantages, lossless data embedding enables novel elegant applications, such as lossless fragile authentication and erasable robust watermarking[3].

Applications that benefit from lossless data embedding include the whole spectrum of fragile watermarking, such as authentication watermarks or watermarks protecting the image integrity. A classical authentication watermarking scheme starts with dividing the image or its blocks into two parts – the part that carries the majority of the perceptual information and an "unimportant" part that can be randomized without causing perceptible artifacts. The perceptually important part is then hashed and the hash is inserted into the "unimportant" part. The Wong's scheme[10] is an example of such as scheme. In this scheme, the perceptually important part consists of the 7 most significant bits, while the unimportant part is formed by the least significant bit-plane. However, this and similar approaches can be reformulated as simply decreasing the information content of the image and inserting or attaching the hash to the modified image as is done in pure cryptographic authentication methods.

The lossless data embedding enables hash insertion while retaining the information content of the image in its entirety. This is important for customers who are overly concerned with the quality of their images after information has been embedded. Some customers are simply so emotionally attached to their images that no argument about invisibility of the embedding artifacts is convincing enough. Lossless embedding techniques simply close this issue because the original data can be restored without any loss of information. This is especially true for medical images, military images, such as satellite and reconnaissance images, and images used in the court as evidence. Actually, the lossless authentication watermarks are currently being incorporated as an integrity protection mechanism into the ISSE military guard.

Another application that would clearly benefit from the proposed lossless techniques is integrity protection watermark embedded inside the digital camera for imaging hardware used by forensic personnel[11,12,13]. Establishing the integrity of evidence throughout the investigation is of paramount importance. Authentication watermarks embedded by a watermarking chip inside the digital camera have been proposed in the past. However, because the authentication

process invariably modifies the image, today, the legal problems associated with watermarking prevent wider spread of watermarking technology. The removable lossless authentication watermark provides an elegant solution to this sensitive issue.

7. CONCLUSIONS AND FUTURE DIRECTIONS

One common drawback of virtually all image data embedding methods is the fact that the original image is inevitably distorted by some small amount of noise due to data embedding itself. This distortion typically cannot be removed completely due to quantization, bit-replacement, or truncation at the grayscales 0 and 255. Although the distortion is often quite small, it may not be acceptable for medical imagery (for legal reasons), for military images inspected under unusual viewing conditions (after filtering or extreme zoom), and for imagery used as evidence in the court. In this paper, we introduced two general methodologies for high-capacity data embedding that we call lossless in the sense that after the embedded information is extracted from the watermarked image, we can revert to the exact copy of the original image.

For uncompressed image formats, such as the BMP or the TIFF format, we recommend so called RS method that is a fragile high-capacity data embedding technique based on embedding message bits in the status of small groups of pixels. For JPEG images, we described three different methods for lossless data embedding while investigating their capacity and distortion. The first method is based on compression of LSBs of a selected quantized DCT coefficient of selected blocks. The second and third methods use a special trick to preprocess the image to allow trivially lossless data embedding. Finally, we describe several elegant methods for the palette formats, such as GIF or PNG. The paper is closed with outlining the advantages of lossless data embedding and some applications, such as lossless authentication.

ACKNOWLEDGEMENTS

The work on this paper was partially supported by Air Force Research Laboratory, Air Force Material Command, USAF, under a research grant number F30602-00-1-0521 and partially by the AFOSR grant No. F49620-01-1-0123. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation there on. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of Air Force Research Laboratory, the Air Force Office of Scientific Research, or the U. S. Government.

REFERENCES

1. Fridrich, J., Goljan, M., Du, R., “Invertible Authentication”, *Proc. SPIE, Security and Watermarking of Multimedia Contents*, San Jose, California, 2001.
2. Fridrich, J., Du, R., Long, M., “Steganalysis of LSB Encoding in Color Images”, *Proc. ICME 2000*, New York City, New York, 2000.
3. Goljan, M., Fridrich, J., and R. Du, “Distortion-free Data Embedding for Images”, *The 4th Information Hiding Workshop*, Pittsburgh, Pennsylvania, April 25–27, 2001.
4. Fridrich, J., Goljan, M., Du, R., “Invertible Authentication Watermark for JPEG Files”, *Proc. ITCC*, Las Vegas, Nevada, April, 2001.
5. Fridrich, J., Goljan, M., Du, R., “Reliable Detection of LSB Steganography in Grayscale and Color Images”, *Proc. ACM Special Session on Multimedia Security and Watermarking*, Ottawa, Canada, 2001.
6. Honsinger, C. W., Jones, P., Rabbani, M., Stoffel, J. C., “Lossless Recovery of an Original Image Containing Embedded Data”, US Patent application, Docket No: 77102/E–D, 1999.
7. Honsinger, C. W., “A Robust Data Hiding Technique Based on Convolution with a Randomized Phase Carrier”, *Proc. of PICS'00*, Portland, Oregon, March, 2000.
8. Macq, B., “Lossless Multiresolution Transform for Image Authenticating Watermarking”, *Proc. of EUSIPCO*, Tampere, Finland, September, 2000.
9. Sayood, K.: *Introduction to Data Compression*, Morgan Kaufmann Publishers, San Francisco, California, pp. 87−94, 1996.
10. Wong, P. W., "A watermark for image integrity and ownership verification", *Proc. IS&T PIC,* Portland, Oregon, 1998.
11. DSS system by Kodak: www.kodak.com/US/en/digital/software/imageAuthentication/ (link to white paper)
12. IAS (Image Authentication System) system from Epson: www.epson.co.uk/options/imaging/ias.htm
13. Veridata from Signum Technologies: www.signumtech.com/new/template4.asp?pageID=3&appID=3#what