Surmounting the Effects of Lossy Compression on Steganography

Daniel L. Currie, III
Fleet Information Warfare Center
2555 Amphibious Drive
NAB Little Creek
Norfolk, VA 23521-3225
currie@msn.comi

Cynthia E. Irvine
Computer Science Department
Code CS/Ic
Naval Postgraduate School
Monterey, CA 93943-5118
irvine@cs.nps.navy.mil

Abstract

Steganographic techniques can be used to hide data within digital images with little or no visible change in the perceived appearance of the image and can be exploited to export sensitive information. Since images are frequently compressed for storage or transmission, effective steganography must employ coding techniques to counter the errors caused by lossy compression algorithms. The Joint Photographic Expert Group (JPEG) compression algorithm, while producing only a small amount of visual distortion, introduces a relatively large number of errors in the bitmap data. It is shown that, despite errors caused by compression, information can be steganographically encoded into pixel data so that it is recoverable after JPEG processing, though not with perfect accuracy.

1. Introduction

Two techniques are available to those wishing to transmit secrets using unprotected communications media. One is cryptography, where the secret is scrambled and can be reconstituted only by the holder of a key. When cryptography is used, the fact that the secret was transmitted is observable by anyone. The second method is steganography [1]. Here the secret is encoded in another message in a manner such that, to the casual observer, it is unseen. Thus, the fact that the secret is being transmitted is also a secret.

Widespread use of digitized information in automated information systems has resulted in a renaissance for steganography. Information which provides the ideal vehicle for steganography is that which is stored with an accuracy far greater than necessary for the data's use and display. Image, Postscript, and audio files are among those that fall into this category, while text, database, and executable code files do not.

It has been demonstrated that a significant amount of information can be concealed in bitmapped image files with little or no visible degradation of the image[4]. This process, called steganography, is accomplished by replacing the least significant bits in the pixel bytes with the data to be hidden. Since the least significant pixel bits contribute very little

1. The term steganography derives from a method of hidden writing discussed by Trimetheus in his three-volume Steganographia, published in 1499 [3].

to the overall appearance of the pixel, replacing these bits often has no perceptible effect on the image. To illustrate, consider a 24 bit pixel which uses 8 bits for each of the red, green, and blue color channels. The pixel is capable of representing 2^{24} or 16,777,216 color values. If we use the lower 2 bits of each color channel to hide data (Figure 1), the maximum change in any pixel would be 2^6 or 64 color values; a minute fraction of the whole color space. This small change is invisible to the human eye. To continue the example, an image of 735 by 485 pixels could hold 735*485 * 6 bits/pixel * 1byte/8 bits = 267,356 bytes of data.

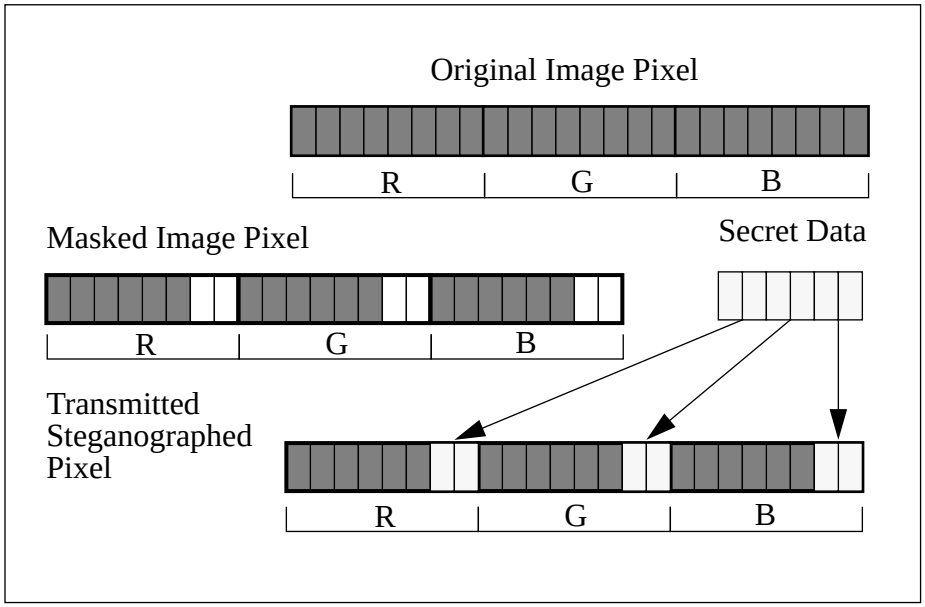

Figure 1:

Kurak and McHugh [4] show that it is even possible to embed one image inside another. Further, they assert that visual inspection of an image prior to its being downgraded is insufficient to prevent unauthorized flow of data from one security level to a lower one.

A number of different formats are widely used to store imagery including BMP, TIFF, GIF, etc. Several of these image file formats “palletize” images by taking advantage of the fact that the color veracity of the image is not significantly degraded to the human observer by drastically reducing the total number of colors available. Instead of over 16 million possible colors, the color range is reduced and stored in a table. Each pixel, instead of containing a precise 24-bit color, stores an 8-bit index into the color table. This reduces the size of the bitmap by 2/3. When the image is processed for display by a viewer such as “xv” [1], the indices stored at the location of each pixel are used to obtain the colors to be displayed from the color table. It has been demonstrated that steganography is ineffective

when images are stored using this compression algorithm[2]. Difficulty in designing a general-purpose steganographic algorithm for palletized images results from the following factors: a change to a “pixel” results in a different index into the color table, which could result in a dramatically different color, changes in the color table can result in easily perceived changes to the image, and color maps vary from image to image with compression choices made as much for aesthetic reasons as for the efficiency of the compression.

Despite the relative ease of employing steganography to covertly transport data in an uncompressed 24-bit image, lossy compression algorithms based on techniques from digital signal processing, which are very commonly employed in image handling systems, pose a severe threat to the embedded data. An excellent example of this is the ubiquitous Joint Photographic Experts Group (JPEG) [7][5] compression algorithm which is the principle compression technique for transmission and storage of images used by government organizations. It does a quite thorough job of destroying data hidden in the least significant bits of pixels. The effects of JPEG on image pixels and coding techniques to counter its corruption of steganographically hidden data are the subjects of this paper.

2. JPEG Compression

JPEG has been developed to provide efficient, flexible compression tools. JPEG has four modes of operation designed to support a variety of continuous-tone image applications. Most applications utilize the Baseline sequential coder/decoder which is very effective and is sufficient for many applications.

JPEG works in several steps. First the image pixels are transformed into a luminance/ chrominance color space [6] and then the chrominance component is downsampled to reduce the volume of data. This downsampling is possible because the human eye is much more sensitive to luminance changes than to chrominance changes. Next, the pixel values are grouped into 8x8 blocks which are transformed using the discrete cosine transform (DCT). The DCT yields an 8x8 frequency map which contains coefficients representing the average value in the block and successively higher-frequency changes within the block. Each block then has its values divided by a quantization coefficient and the result rounded to an integer. This quantization is where most of the loss caused by JPEG occurs. Many of the coefficients representing higher frequencies are reduced to zero. This is acceptable since the higher frequency data that is lost will produce very little visually detectable change in the image. The reduced coefficients are then encoded using Huffman coding to further reduce the size of the data. This step is lossless. The final step in JPEG applications is to add header data giving parameters to be used by the decoder.

3. Stego Encoding Experiments

As mentioned before, embedding data in the least significant bits of image pixels is a simple steganographic technique, but it cannot survive the deleterious effects of JPEG. To investigate the possibility of employing some kind of encoding to ensure survivability of

embedded data it is necessary to identify what kind of loss/corruption JPEG causes in an image and where in the image it occurs.

At first glance, the solution may seem to be to look at the compression algorithm to try to predict mathematically where changes to the original pixels will occur. This is impractical since the DCT converts the pixel values to coefficient values representing 64 basis signal amplitudes. This has the effect of spatially "smearing" the pixel bits so that the location of any particular bit is spread over all the coefficient values. Because of the complex relationship between the original pixel values and the output of the DCT, it is not feasible to trace the bits through the compression algorithm and predict their location in the compressed data.

Due to the complexity of the JPEG algorithm an empirical approach to studying its effects is called for. To study the effects of JPEG, 24 bit Windows BMP format files were compressed, decompressed, with the resulting file saved under a new filename.

Figure 2:

The BMP file format was chosen for its simplicity and widespread acceptance for image processing applications. For the experiments, two photographs, one of a seagull and one of a pair of glasses (Figure 2 and Figure 3), were chosen for their differing amount of detail and number of colors. JPEG is sensitive to these factors. Table 1 below shows the results of a byte by byte comparison of the original image files and the JPEG processed versions, normalized to 100,000 bytes for each image. Here we see that the seagull picture has fewer than half as many errors in the most significant bits (MSB) as the glasses picture. While the least significant bits (LSB) have an essentially equivalent number of errors.

Figure 3:

	MSB 8	7	6	5	4	3	2	LSB 1
Glasses	744	4032	10247	21273	33644	42327	27196	48554
Seagull	257	991	2821	7514	15039	29941	41593	46640

Table 1:

Table 2 shows the Hamming distance (number of differing bits) between corresponding pixels in the original and JPEG processed files normalized to 100,000 pixels for each image. Again, the seagull picture has fewer errors.

	1-3	4-6	7-9	10-12	13-15	16-18	19-21	22-24
Glasses	15581	37135	30337	11976	2205	172	4	0
Seagull	24188	38710	17564	4631	409	43	1	0

Table 2:

Given the information in Table 1, it is apparent that data embedded in any or all of the lower 5 bits would be corrupted beyond recognition. Attempts to embed data in these bits and recover it after JPEG processing showed that the recovered data was completely garbled by JPEG.

Since a straightforward substitution of pixel bits with data bits proved useless, a simple coding scheme to embed one data bit per pixel byte was tried. A bit was embedded in the lower 5 bits of each byte by replacing the bits with 01000 to code a 0 and 11000 to code a 1. On decoding, any value from 00000 to 01111 would be decoded as a 0 and 10000 to 11111 as a 1. The hypothesis was that perhaps JPEG would not change a byte value by more than 7 in an upward direction and 8 in a downward direction or, if it did, it would make drastic changes only occasionally and some kind of redundancy coding could be used to correct errors. This approach failed. JPEG is indiscriminate about the amount of change it makes to byte values and produced enough errors that the hidden data was unrecognizable.

The negative results of the first few attempts to embed data indicated that a more subtle approach to encoding was necessary. It was noticed that, in a JPEG processed image, the pixels which were changed from their original appearance were similar in color to the original. This indicates that the changes made by JPEG, to some extent, maintain the general color of the pixels. To attempt to take advantage of this, a new coding scheme was devised based on viewing the pixel as a point in space (Figure 4) with the three color channel values as the coordinates.

The coding scheme begins by computing the distance from the pixel to the origin (0,0,0). Then the distance is divided by a number and the remainder (r = distance mod n) is found. The pixel value is adjusted such that its remainder is changed to a number corresponding to the bit value being encoded. Qualitatively, this means that the length of the vector representing the pixel's position in three-dimensional RGB color space is modified to encode information. Because the vector's direction is unmodified, the relative sizes of the color channel values are preserved.

Suppose we choose an arbitrary modulus of 42. When the bit is decoded, the distance to origin will be computed and any value from 21 to 41 will be decoded as a 1 and any value from 0 to 20 will be decoded as a 0. So we want to move the pixel to a middle value in one of these ranges to allow for error introduced by JPEG. In this case, the vector representing the pixel would have its length modified so that the remainder is 10 to code a 0 or a 31 to code a 1. It was hoped that JPEG would not change the pixel's distance from the origin by more than 10 in either direction thus allowing the hidden information to be correctly decoded.

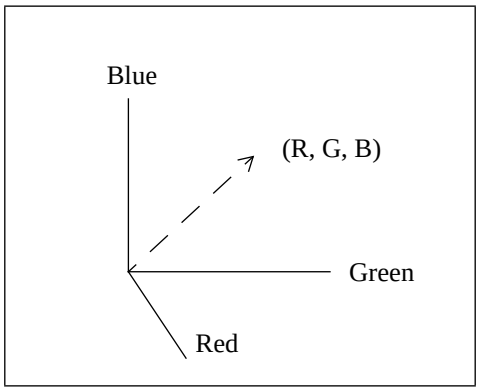

Figure 4:

For example, given a pixel (128, 65, 210) the distance to the origin would be computed: $d = \sqrt{128^2 + 65^2 + 210^2} = 254.38$. The value of d is rounded to the nearest integer. Next we find $r \equiv \mathrm{d} \bmod 42$, which is 2. If we are coding a 0 in this pixel, the amplitude of the color vector will be increased by 8 units to an ideal remainder of 10 (d = 262) and moved down 13 (d = 241) units to code a 1. Note that the maximum displacement any pixel would suffer would be 21. Simple vector arithmetic permits the modified values of the red, green, and blue components to be computed. The results of using this encoding are described in the next section.

Another similar technique is to apply coding to the luminance value of each pixel in the same way as was done to the distance from origin. The luminance, y, of a pixel is computed as y = 0.3R + 0.6G + 0.1B [6]. Where R, G, and B are the red, green, and blue color values respectively. This technique appears to hold some promise since the number of large changes in the luminance values caused by JPEG is not a high as with the distance from origin. One drawback of this technique is that the range of luminance value is from 0 to 255 whereas the range of the distance from origin is 0 to 441.67.

4. Discussion of Experiments

With ordinary embedding techniques which simply replace image bits with data bits one is forced to use an enormous amount of redundancy to achieve a suitably low error rate after JPEG compression. Also, since the lowest few bits are so badly scrambled by JPEG, higher order bits must be used which increases the visible distortion of the image. This is contrary to steganography's goal of being a covert communication channel.

The distance from origin technique results in a lower error rate, but requires the addition of repetitive redundancy to even attempt to achieve a reasonable error rate. Specifically, the average displacement by JPEG of pixels in the seagull picture with respect to the origin is 2.36. But, if a modulus of 62 is assumed, the number of pixels displaced by 30 (enough to cause an error) is 3100 or 0.8678%. Given this figure and applying triple redundancy in embedding data (i.e. embed each bit three times in a row) yields a theoretical error rate of $(.008678)^3 + 3(0.008678)^2(0.991322) = 0.000225$ per bit or $1 - (1 - 0.000225)^8 = 0.001799$ per byte.

Unfortunately, these calculations do not take into account the peculiar and obstreperous nature of JPEG processing. JPEG produces the most errors in areas of an image where the pixel values vary the most, i.e. color transients produce local errors the number of which is proportional to the amount of the transients. The process of embedding data invariably causes a different pattern and amount of color transients. So embedding data actually *causes* more errors in the area where data is embedded.

Despite the active way in which the JPEG algorithm garbles embedded data we were able to demonstrate a recovery rate of approximately 30% of embedded data using RGB coding coupled with multiple redundancy coding.

5. Summary

As an anti-steganography technique, JPEG is very effective. In order to achieve anything approaching an acceptable error rate, a great deal of redundancy must be applied. With the distance-to-origin and luminance modulo coding techniques the error rate can be brought down. But these techniques must be coupled with multiple redundancy to lower the error rate. Although the amount of data which can be hidden may be relatively small compared to the size of the image, the security threat that steganography poses cannot be completely eliminated by application of a transform-based lossy compression algorithm.

This paper describes preliminary research. Future work will examine the internal details of the JPEG algorithm to determine the effects of the discrete cosine transform and the quantization factors on the information content of images with and without steganographic embedding. We are also investigating potential techniques to detect steganography in images.

6. Acknowledgements

The authors would like to thank Hannelore Campbell, Hal Fredericksen, and David Wootten for many helpful ideas, criticisms, and discussions.

References

1. Bradley, J., XV - Version 3.00, Rev. 3/30/93.
2. Cha, S.D., Park, G.H., and Lee, H.K., “A Solution to the On-Line Image Downgrading Problem,” in *Proceedings of the Eleventh Annual Computer Security Applications Conference*, New Orleans, LA, pp. 108-112, December 1995.
3. Kahn, D., *The Codebreakers*, MacMillan Company, 1967.
4. Kurak, C., McHugh, J., “A Cautionary Note on Image Downgrading,”in *Proceedings of the 8th Annual Computer Security Applications Conference*, pp 153-59.

5.. Wallace, Gregory K., “The JPEG Still Picture Compression Standard”, *Communications of the ACM*, Vol. 34, No. 4, April 1991.

6. Pennebaker, William B., Mitchell, Joan L., *JPEG Still Image Compression Standard*, Van Nostrand Reinhold, New York, 1993.
7. *Joint Photographic Experts Group (JPEG) Compression for the National Imagery Transmission Format Standard*, MIL-STD-188-198A, December 1993.