The International Journal Of Engineering And Science (IJES)
|| Volume || 3 || Issue || 5 || Pages || 56-61 || 2014 ||
ISSN (e): 2319 – 1813 ISSN (p): 2319 – 1805

New Steganographic Technique That Can Reliably Detect Modifications in Digital Images Based on JPEG Compatibility

[1,] Asst.Prof. Dipayan Kumar Ghosh, [2,] Namita Ghosh

[1,] Assistant Professor in Computer Science & Engineering (CSE) Depratment
Calcutta Institute of Technology (CIT) Uluberia, Howrah – 711316, West Bengal, India.
[2,] MCA, Haldia Institute of Technology (HIT) Midnapore, West Bengal, India.

ABSTRACT

In this paper, we introduce a new forensic tool that can reliably detect modifications in digital images, such as distortion due to steganography and watermarking, in images that were originally stored in the JPEG format. The JPEG compression leaves unique fingerprints and serves as a "fragile watermark" enabling us to detect changes as small as modifying the LSB of one randomly chosen pixel. The detection of changes is based on investigating the compatibility of 8×8 blocks of pixels with JPEG compression with a given quantization matrix. The proposed steganalytic method is applicable to virtually all steganographic and watermarking algorithms with the exception of those that embed message bits into the quantized JPEG DCT coefficients. The method can also be used to estimate the size of the secret message and identify the pixels that carry message bits. As a consequence of our steganalysis, we strongly recommend avoiding using images that have been originally stored in the JPEG format as cover-images for spatial-domain steganography.

KEYWORDS: *Steganography, steganalysis, JPEG*

Date of Submission: 07 May 2014 Date of Publication: 20 May 2014

I. INTRODUCTION

Steganography is the art of secret communication. Its purpose is to hide the very presence of communication as opposed to cryptography whose goal is to make communication unintelligible to those who do not posses the right keys[1]. Digital images, videos, sound files, and other computer files that contain perceptually irrelevant or redundant information can be used as "covers" or carriers to hide secret messages. After embedding a secret message into the cover-image, a so-called stego-image is obtained. It is important that the stego-image does not contain any easily detectable artifacts due to message embedding. A third party could use such artifacts as an indication that a secret message is present. Once this message detection can be reliably achieved, the steganographic tool becomes useless.

Obviously, the less information is embedded into the cover-image, the smaller the probability of introducing detectable artifacts by the embedding process. Another important factor is the choice of the cover-image. The selection is at the discretion of the person who sends the message. The sender should avoid using cover-images that would be easy to analyze for presence of secret messages. For example, one should not use computer art, charts, images with large areas of uniform color, images with only a few colors, and images with a unique semantic content, such as fonts. Although computer-generated fractal images may seem as good covers[6] because of their complexity and irregularity, they are generated by strict deterministic rules that may be easily violated by message embedding[3].

Scans of photographs or images obtained with a digital camera contain a high number of colors and are usually recommended and considered safe for steganography. Some steganographic experts recommend grayscale images as the best cover-images[2].

There are essentially three types of image formats: raw, uncompressed formats (BMP, PCX), palette formats (GIF), and lossy compressed formats (JPEG, Wavelet, JPEG2000). Only few current steganographic programs offer the capability to embed messages directly in the JPEG stream. It is a difficult problem to devise a steganographic method that would hide messages in the JPEG stream in a secure manner while keeping the capacity practical. Far more programs use the BMP, PCX, or the GIF palette-based format. The GIF format is a difficult environment for secure steganography with reasonable capacity[3,7]. Also, most steganographic techniques for GIFs implemented in current software products prioritize capacity over security and are thus relatively easy to detect[4,5]. The raw formats, such as BMP, offer the highest capacity and best overall security. In this paper, we demonstrate that even 24-bit images or grayscale 8-bit images may actually be *extremely* poor candidates for cover-images if they were initially acquired as JPEG images and later decompressed to a lossless format. In fact, it is quite reasonable to expect that most casual users of steganographic programs will use scanned images or images from a digital camera that were originally stored in the JPEG format due to its efficiency in data storage.

All steganographic methods strive to achieve the minimal amount of distortion in order to minimize the likelihood of introducing any visible artifacts. Consequently, if the cover-image, was initially stored in the JPEG format, the act of message embedding will *not* erase the characteristic structure created by the JPEG compression and one can still easily determine whether or not a given image has been stored as JPEG in the past. Actually, unless the image is too small, one can reliably recover even the values of the JPEG quantization table by carefully analyzing the values of DCT coefficients in all 8×8 blocks. After message embedding, however, the cover-image will become (with a high probability) incompatible with the JPEG format in the sense that it may be possible to prove that a particular 8×8 block of pixels could not have been produced by JPEG decompression of any block of quantized coefficients. This finding provides strong evidence that the block has been modified. It is highly suspicious to find an image stored in a lossless format that bears a strong fingerprint of JPEG compression, yet is not fully compatible with any JPEG compressed image. This can be interpreted as evidence for steganography.

By checking the JPEG compatibility of every block, we can potentially detect messages as short as one bit. And the steganalytic method will work for virtually any steganographic or watermarking method, not just the LSB embedding! Indeed, in our experiments, we have found out that even one randomly selected pixel whose gray level has been modified by one can be detected with very high probability. For longer messages, one can even attempt to estimate the message length and its position in the image by determining which 8×8 blocks are incompatible with JPEG compression. It is even possible to analyze the image and estimate the likely candidate for the cover-image or its blocks (the "closest" JPEG compatible image/block). This way, we may be able to identify individual pixels that have been modified. All this indicates that an extremely serious information leakage from the steganographic method can occur and thus completely compromise the steganographic channel.

In this paper, we elaborate on the idea presented in the previous paragraph. In Section 2, we describe an algorithm that can decide if a given 8×8 block of pixels is compatible with JPEG compression with a given quantization matrix. Appendix A contains details on how the quantization matrix can be estimated from the image. Throughout the paper, we point out some limitations of the proposed steganalytic technique and finally, in Section 3, we conclude the paper and outline future research directions. As a consequence of this research, we strongly urge users of steganographic programs to avoid images previously stored in the JPEG format as cover-images.

II. STEGANALYSIS BASED ON JPEG COMPATIBILITY

Although in this paper we explain the technique on grayscale images, it can be extended to color images in a straightforward manner. We start with a short description of the JPEG compression algorithm. In JPEG compression, the image is first divided into disjoint blocks of 8×8 pixels. For each block B_{orig} (with integer pixel values in the range 0–255), the discrete cosine transform (DCT) is calculated, producing 64 DCT coefficients. Let us denote the i-th DCT coefficient of the k-th block as $d_k(i)$, $0 \le i \le 64$, $k = 1, \ldots, T$, where T is the total number of blocks in the image. In each block, all 64 coefficients are further quantized to integers $D_k(i)$ using the JPEG quantization matrix Q

$$D_k(i) = integer\_round\left(\frac{d_k(i)}{Q(i)}\right).$$

The quantized coefficients $D_k(i)$ 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 decompression works in the opposite order. The JPEG bit-stream is decompressed using the Huffman coder and the quantized DCT coefficients $D_k(i)$ are multiplied by $Q(i)$ to obtain DCT coefficients QD_k, $QD_k(i) = Q(i)D_k(i)$ for all k and i. Then, the inverse DCT is applied to QD_k and the result is rounded to integers in the range 0–255

$$B = [B_{raw}] \text{ , where } B_{raw} = DCT^{-1}(QD) , \tag{1}$$

and $[x]$=*integer_round*(x) for $0 \le x \le 255$, $[x]=0$ for $x < 0$, and $[x]=255$ for $x > 255$. In the last expression, we dropped the block index k to simplify the notation. We note that because the JPEG compression is lossy, in general B_{orig} may not be equal to B.

If the block B has no pixels saturated at 0 or 255, we can write in the L^2 norm

$$||B_{raw}–B||^2 \le 16, \tag{2}$$

because $|B_{raw}(i)–B(i)|\le 1/2$ for all $i = 1, \ldots, 64$ due to rounding.

Suppose that we know the quantization matrix Q (see Appendix A). Our steganalytic technique is based on the following question:

> *Given an arbitrary* 8×8 *block of pixel values B, could this block have arisen through the process of JPEG decompression with the quantization matrix Q?*

Denoting $QD'=DCT(B)$, we can write using the Parserval's equality

$$\left\|B - B_{raw}\right\|^2 = \left\|DCT(B) - DCT(B_{raw})\right\|^2 = \left\|QD' - QD\right\|^2 \le 16 \;.$$

On the other hand, we can find a lower estimate for the expression $||QD'–QD||^2$ by substituting for $QD(i)$ the closest integer multiple of $Q(i)$:

$$16 \ge \left\|QD' - QD\right\|^2 \ge \sum_{i=1}^{64} \left|QD'(i) - Q(i)\,round\left(\frac{QD'(i)}{Q(i)}\right)\right| = S \;. \tag{3}$$

The quantity S can be calculated from the block B provided the quantization matrix Q is known. If S is larger than 16, we can conclude that the image block B is not compatible with JPEG compression with the quantization matrix Q. We reiterate that this is true only for blocks that do not have pixels that are saturated at 0 or 255. Indeed, the estimate (2) may not hold for blocks that have saturated pixels because the rounding at 0 and 255 can be much larger than 1/2.

If, for a given block B with unsaturated pixels, S is smaller than 16, the block B may or may not be JPEG compatible. Let $q_p(i)$, p=1, … , be integer multiples of $Q(i)$ that are closest to $QD(i)$ ordered by their distance from $QD(i)$ (the closest multiple is $q_1(i)$). In order to decide whether or not a given block B is compatible with JPEG compression with quantization table Q, we need to inspect all 64-tuples of indices $\{p(1), \ldots, p(64)\}$ for which

$$S = \sum_{i=1}^{64} \left|QD'(i) - q_{p(i)}(i)\right| \le 16 \tag{4}$$

and check if

$$B=[DCT^{-1}(QD)], \text{ where } QD(i)= q_{p(i)}(i). \tag{5}$$

If, for at least one set of indices $\{p(1), \ldots, p(64)\}$, the expression (5) is satisfied, the block B is JPEG compatible, otherwise it is not.

The number of 64-tuples $\{p(1), \ldots, p(64)\}$ satisfying expression (4) is always finite but it rapidly increases with increasing JPEG quality factor. For quality factors higher than 95, a large number of quantization factors $Q(i)$ become 1 or 2, and the total number of combinations of all 64 indices becomes too large to handle. We performed our experiments in Matlab on a Pentium II computer with 128MB memory. Once the quality factor exceeded 95, the running time became too long because Matlab ran out of memory and had to access the hard disk. We acknowledge this complexity increase as a limitation of our approach. In the future, we would like to develop a better and faster algorithm for testing JPEG compatibility for high quality compression factors.

Description of the algorithm:

1. Divide the image into a grid of 8×8 blocks, skipping the last few rows or columns if the image dimensions are not multiples of 8.
2. Arrange the blocks in a list and remove all saturated blocks from the list (a block is saturated if it has at least one pixel with a gray value 0 or 255). Denote the total number of blocks as T.
3. Extract the quantization matrix Q from all T blocks as described in Appendix A. If all the elements of Q are ones, the image was not previously stored as JPEG and our steganalytic method does not apply (exit this algorithm). If more than one plausible candidate exists for Q, the steps 4–6 need to be carried out for all candidates and the results that give the highest number of JPEG compatible blocks will be accepted as the result of this algorithm.
4. For each block B calculate the quantity S (see equation (3)).
5. If $S>16$, the block B is not compatible with JPEG compression with quantization matrix Q. If $S\leq16$, for each DCT coefficient QD_i' calculate the closest multiples of $Q(i)$, order them by their distance from QD_i', and denote them $q_p(i)$, $p=1, \ldots$. For those combinations, for which the inequality (4) is satisfied, check if expression (5) holds. If, for at least one set of indices $\{p(1), \ldots, p(64)\}$ the expression (5) is satisfied, the block B is JPEG compatible, otherwise it is not.
6. After going through all T blocks, if no incompatible JPEG blocks are found, the conclusion is that our steganalytic method did not find any evidence for presence of secret messages. If, on the other hand, there are some JPEG incompatible blocks, we can attempt to estimate the size of the secret message, locate the message-bearing pixels, and even attempt to obtain the original cover image before secret message embedding started.
7. If all blocks are identified as JPEG incompatible or if the image does not appear to be previously stored as JPEG, we should repeat the algorithm for different 8×8 divisions of the image (shifted by 0 to 7 pixels in the x and y directions). This step may be necessary if the cover image has been cropped prior to message embedding.

III. CONCLUSIONS AND FUTURE EFFORT

In this paper, we describe a new steganographic technique that can reliably detect modifications in digital images, such as those due to steganography and watermarking, as long as the original image (the cover-image) has been previously stored in the JPEG format. The steganalytic technique starts with extracting the JPEG quantization matrix by carefully inspecting the clusters of DCT coefficients in all 8×8 blocks. Then, each block is analyzed if its pixel values are truncated values of an inverse DCT transform of a set of coefficients quantized with the extracted quantization matrix. A simple necessary condition is derived that makes the analysis computationally feasible. If the corresponding quantized DCT coefficients are found, the block is termed compatible, otherwise it is not.

The steganalytic technique will work for all steganographic methods, except the methods that embed information directly into the compressed JPEG stream. It has a potential to detect changes as small as one pixel. By inspecting the closest compatible JPEG block, we can even attempt to locate the pixels that have been modified.

As a consequence of this research, we strongly urge the users of steganographic programs to avoid using images that have been previously stored in the JPEG format as cover-images for steganography. The JPEG compression imprints a unique fingerprint on the image and this fingerprint can be carefully utilized for steganalysis as shown in this paper. If no other images are available, try to rescale or resample the image to slightly smaller dimensions to wipe out the JPEG fingerprint. Applying filters, adjusting contrast, or slightly rotating the image may also help in reducing the JPEG fingerprint.

The proposed steganalytic method has some limitations that we would like to address in the future. First, the necessary condition derived for unsaturated blocks (3) does not hold for blocks that have some black or white pixels. The study of JPEG compatibility of blocks that have some saturated pixels appears to be a difficult challenge that will be pursued in the future.

Another limitation of our technique is the rapidly increasing computational complexity with increasing JPEG quality factor. The running time became unacceptably large for quality factors larger than 95. This is due to the large number of candidate JPEG blocks that need to be inspected for compatibility. We believe that this complexity problem could be resolved by deriving a set of other necessary conditions similar to (3) to decrease the number of possible candidates that need to be inspected for compatibility in Step 5.

APPENDIX A

In this appendix, we show how one can extract the quantization matrix *Q* from a (BMP) image. Keeping the same notation as in the previous section, we first calculate the DCT coefficients $d_k(i)$, $0 \le i \le 64$, $k = 1, \ldots, T$ from all unsaturated 8×8 blocks. For each coefficient *i*, we plot the quantity $E_i(q)$ as a function of $q = 1, 2, \ldots$

$$E_i(q) = \frac{1}{T}\sum_{k=1}^{T}\left| d_k(i) - q \times integer\_round\left(\frac{d_k(i)}{q}\right)\right| .$$

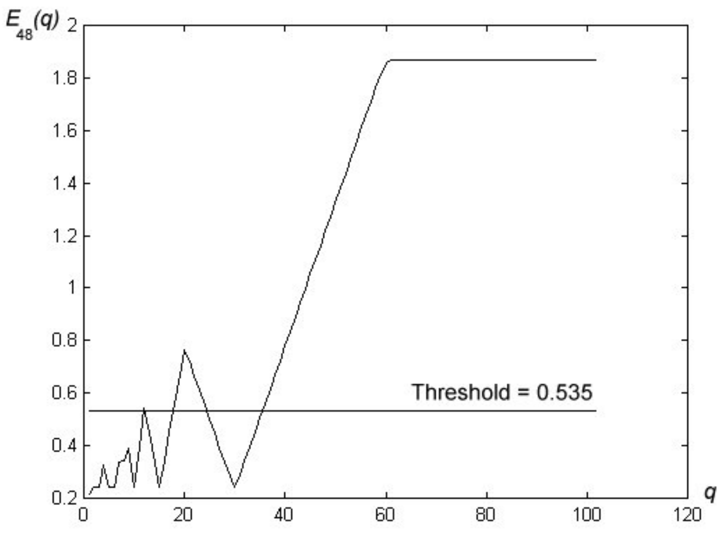

Figure 1 The error $E_{48}(q)$ for the 48-th DCT coefficient (8,6) plotted as a function of the quantization step *q*. Notice the local minima at 30 and all its divisors 15, 10, 6, 5, 3, and 2. The last minimum is 30, which corresponds to the correct quantization step for JPEG quality factor 85.

The quantity $E_i(q)$ measures the compatibility of all *i*-th DCT coefficients in the image with the quantization step *q*. If the image under inspection was indeed previously stored as JPEG, we will observe a drop (local minimum) at the correct quantization value *q* and at all integer divisors of *q*. It is intuitively clear that the correct value of the quantization step should be the largest value *q* at which a local minimum of *E* occurs. As discussed below, this may not, however, be true in all cases.

First of all, it is possible that some "ghost" local minima can occur for values of *q* larger than the correct value. Those minima, however, are significantly larger than the other "correct" minima. By setting a threshold on the largest possible value of a local minimum, we can successfully "filter out" the false minima. In our experiments, we make the thresholod dependent on the DCT coefficient (on index *i*) and calculate its value as $\mu_i+3\sigma_i$, where μ_i and σ_i are the mean and the standard deviation of all local minima of the vector $E_i(q)$, $q=1, \ldots.$

If the vector $E_i(q)$ does not have any local minima except for q=1 (we note that $E_i(1) \leq E_i(q)$ for all q>1), we inspect the difference between $E_i(1)$ and $E_i(2)$ to decide if the quantization step is 1 or 2. For this case, we developed an empirical rule that gives very good results. Based on our experiments, if $E_i(1) < 0.6\, E_i(2)$, we conclude that the quantization step is 1, otherwise it is 2.

This process of finding the quantization steps for each coefficient is still not completely foolproof. For small images, it can happen that, for the quantization step q, all values of the DCT coefficients $d_k(i)$ will be multiples of $2q$. In this case, we have no means to tell if the correct step is $2q$ or q. The probability of this happening decreases with the image size. To address this problem, we inspect the extracted quantization matrix Q for outliers by dividing it by the standard JPEG quantization table. Here, we use the logic that even customized tables (e.g., tables used in JPEG compression in digital cameras) will not significantly deviate from the standard quantization table. Finally, it is always possible to identify candidates for suspicious quantization steps and run the steganalysis for all possible combinations.

REFERENCES

[1]. Andersen, R.J. and Petitcolas, F.A.P., "On the limits of steganography", *IEEE Journal of Selected Areas in Communications, Special Issue on Copyright and Privacy Protection*, **16(**4), pp. 474–481, 1998.

[2]. Aura, T., "Invisible communication", *Proc. of the HUT Seminar on Network Security '95*, Espoo, Finland, November 1995. Telecommunications Software and Multimedia Laboratory, Helsinki University of Technology.

[3]. Fridrich, J. and Du, R., "Secure Steganographic Methods for Palette Images", *Proc. The 3rd Information Hiding Workshop*, September 28–30, Dresden, Germany, LNCS vol. 1768, Springer-Verlag, New York, pp. 47–60, 1999.

[4]. Johnson, N. F. and Jajodia, S., "Steganography: Seeing the Unseen," *IEEE Computer*, February, pp.26–34, 1998.

[5]. Johnson, N. F. and Jajodia, S., "Steganalysis of Images Created Using Current Steganography Software," *Proc. The 2nd Information Hiding Workshop*, Portland, OR, April, LNCS vol.1525, Springer-Verlag, New York, 1998.

[6]. Hastur, H., "Mandelsteg," Software downloadable from http://idea.sec.dsi.uimi.it/pub/security/crypt/codev, 1994.

[7]. Westfeld, A. and Pfitzmann, A., "Attacks on Steganographic Systems", *Proc. The 3rd Information Hiding Workshop*, September 28–30, Dresden, Germany, LNCS vol. 1768, Springer-Verlag, New York, pp. 61–75, 1999.