# Secret Communication Using JPEG Double Compression

Jing-Ming Guo, *Member, IEEE*, and Thanh-Nam Le

***Abstract*—Protecting privacy for exchanging information through the media has been a topic researched by many people. Up to now, cryptography has always had its ultimate role in protecting the secrecy between the sender and the intended receiver. However, nowadays steganography techniques are used increasingly besides cryptography to add more protective layer to the hidden data. In this letter, we show that the quality factor in a JPEG image can be an embedding space, and we discuss the ability of embedding a message to a JPEG image by managing JPEG quantization tables (QTs). In combination with some permutation algorithms, this scheme can be used as a tool for secret communication. The proposed method can achieve satisfactory decoded results with this straightforward JPEG double compression strategy.**

***Index Terms*—Data hiding, JPEG, quality factor.**

## I. Introduction

ALONG with the demand for speed and integrity while exchanging information over the Internet, there is always the need for secrecy. Many works has focused on how to protect private information from being attacked and/or identified. Besides cryptography, steganography is also increasingly used for secure communication [1], [2]. Different from cryptography, the main goal of data hiding is to conceal the hidden data by the carrier media, so that the hidden data is transferred without drawing suspicions. The hiding algorithms are to maintain the natural appearance of the cover media and to keep uninvolved people from even thinking the information exists.

To hide information inside an image, there are several available domains where steganography algorithms exploit such as spatial domain or DCT domain [3]. Among various types of images, JPEG format is a commonly used standard of lossy compression for photographic images. JPEG images can typically gain 30:1 compression ratio with little perceptible loss in image quality. Another advantage of JPEG standard is that the degree of compression can be adjusted, allowing a selectable tradeoff between storage size and image quality.

In this work, a secret communication scheme is proposed, in which the data is doubly protected by both encryption stage and hiding stage. The message to be embedded is first processed by applying some encryption techniques. Herein, the permutation algorithm that requires a pair of numbers as a key is employed to permute the original message. After the encryption stage, the scrambled message is then embedded into a JPEG image by managing different quantization tables. The final result is a JPEG image containing some certain regions with different image quality. The recipient performs reverse steps to extract the information: first extract the pattern of the scrambled message, and then use the key which was shared previously to decipher the message.

In Sections II and III, we first briefly describe the standard JPEG quantization process, and explain how resaving a JPEG image which contains different regions with different image quality can expose the embedded data. The scheme for embedding encrypted data into the image is then introduced. The rest of the paper shows the experimental results and draws the conclusions.

## II. Proposed Hidden Information Embedding/Extracting Scheme

### A. JPEG Image Compression Process Revisited

The standard JPEG is mostly employed to deal with color images in RGB format. The first step is to convert the image from RGB space into luminance/chrominance spaces Y, Cb and Cr. Color space conversion step is followed by the subsampling step, where typically the chrominance channels (Cb and Cr) are subsampled with the rate equal to half of the rate of the Y channel.

Each channel is then partitioned into $8 \times 8$ nonoverlap blocks. The pixel values in the channels are shifted from the range $[0, 255]$ to $[-128, 127]$ for the next step Discrete Cosine Transformation (DCT). DCT is a powerful transformation, since the low-frequency and the high-frequency components are separated after this step (the upper left corner of the $8 \times 8$ block contains low frequency coefficients) [5]. This step allows us to truncate away the high frequency coefficients, by applying specific quantization tables (QTs). A larger coefficients set in a quantization table leads to a higher compression rate, but also reduces the image quality, and vice versa. Different software (like Matlab or Photoshop) and different camera models use different QTs for the same quality. For example, the two QTs in Fig. 1 are different, but they provide the same approximate quality factor of 80. For people who are interested in JPEG scheme, many collected QTs are listed in [6].

### B. The Principle of This Message Hiding Scheme

Consider the case when a set of coefficients $c_1$ is quantized by an amount $q_1$, i.e., a block of size $8 \times 8$ contains DCT coef-

Manuscript received April 27, 2010; revised June 24, 2010; accepted June 25, 2010. Date of current version August 23, 2010. This work was supported by the National Science Council of Taiwan under Contract NSC 98-2221-E-011-097-MY2. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Alex C. Kot.

The authors are with the Department of Electrical Engineering, National Taiwan University of Science and Technology, Taipei, Taiwan (e-mail: jmguo@seed.net.tw; namlt243@gmail.com)

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/LSP.2010.2066110

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 6 | 4 | 4 | 6 | 10 | 16 | 20 | 24 |
| 5 | 5 | 6 | 8 | 10 | 23 | 24 | 22 |
| 6 | 5 | 6 | 10 | 16 | 23 | 28 | 22 |
| 6 | 7 | 9 | 12 | 20 | 35 | 32 | 25 |
| 7 | 9 | 15 | 22 | 27 | 44 | 41 | 31 |
| 10 | 14 | 22 | 26 | 32 | 42 | 45 | 37 |
| 20 | 26 | 31 | 35 | 41 | 48 | 48 | 40 |
| 29 | 37 | 38 | 39 | 45 | 40 | 41 | 40 |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 6 | 4 | 3 | 6 | 9 | 15 | 19 | 22 |
| 4 | 4 | 5 | 7 | 9 | 21 | 22 | 20 |
| 5 | 4 | 6 | 9 | 15 | 21 | 25 | 21 |
| 5 | 6 | 8 | 10 | 19 | 32 | 30 | 23 |
| 6 | 8 | 13 | 21 | 25 | 40 | 38 | 28 |
| 9 | 13 | 20 | 24 | 30 | 39 | 42 | 34 |
| 18 | 24 | 29 | 32 | 38 | 45 | 45 | 37 |
| 27 | 34 | 35 | 36 | 42 | 37 | 38 | 37 |

Fig. 1. To compress image at quality of approximate 80, Photoshop and Matlab employ different quantization tables. Table QT1 is used by Matlab while Table QT2 is used by ACDSee.

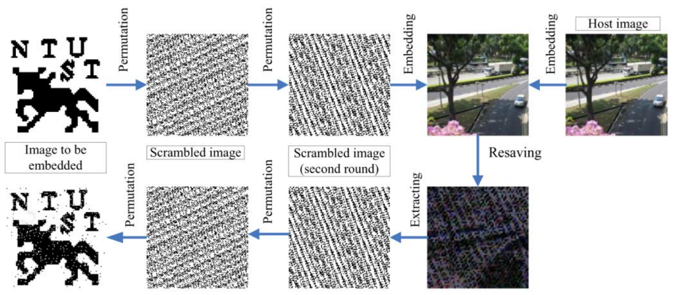

Fig. 2. Embedding and extracting message scheme.

ficients $c_{ij}$ from $c_{11}$ to $c_{88}$ maps to a QT of the same size. Each DCT coefficient is quantized by the corresponding amount in QT and then rounded to the nearest integer (the rounding rule may vary between different schemes):

$$\hat{c}_{ij} = round\left(\frac{c_{ij}}{q_{ij}}\right). \tag{1}$$

If that DCT set is recalculated ($\hat{c}_1 = \hat{c}_{ij} \times q_{ij}$) and then the $\hat{c}_1$ set is quantized a second time by an amount $q_2$, which after reconstruction gives the DCT coefficients set $c_2$. Except in the case of $q_2 = 1$ (which means no more quantization), the difference between $c_2$ and $c_1$ will be minimal only when $q_2 = q_1$. This characteristic was investigated in [4]. In other words, if $c_1$ was previously quantized by a value $q_0$, in which $q_0 > q_1$, means that it was processed with lower quality, and then the difference defined by (2) reaches minimum value as $q_2 = q_1$, if we consider the difference in (2) as a function of $q_2$.

$$difference = \sum_{i,j} \left\| c_1^{ij} - c_2^{ij} \right\|^2 (i, j = 1, \ldots, 8). \tag{2}$$

Besides, the difference also reaches a local minimum as $q_2$ reaches $q_0$. To embed a secret message into a JPEG image, some specific regions of the image are compressed with lower quality $q_0$. The regions with lower quality are employed to carry the hidden information, and an embedded image is used as a media for secret communication.

*1) Encryption Stage:* Fig. 2 illustrates the stages in our proposed scheme. The image to be embedded is first passed through an encryption stage. Here we simply apply an automorphism algorithm [7], [8] to permute the pixels in the hidden message. In fact, any encryption algorithm can be applied in this stage, provided that the algorithm itself and the key are strong enough. An image, after applying some modulo operation, becomes a random pattern. For example, after applying this operator with parameter $w = 2$ on the original image 66 times ($n = 66$), we have the scrambled image like the image next to the original one; $w$ is the parameter for changing the divisor in this modulo operation and is considered the key for decryption. To recover the original image, one has to know the pair $(w, n)$. In this case, to decode we apply the same operator with $w = 2$ and $n = 126$ (with this specific setup, applying the operation 192 times yields the original image). To be even more secure, in practice this step can be repeated with several rounds, and different sets of $w$ and $n$ can be applied. Hence, the key that the recipient needs to reconstruct hidden image is the set of those $w$ and $n$ values.

*2) Embedding to JPEG Cover Image Stage:* Embedding the encrypted pattern to a natural image is the way to camouflage the pattern. Each pixel of the hidden image is embedded to a specific region of a JPEG image by managing the quality factors (actually the quantization tables).

The embedding stage is as follows. Suppose the secret message $\boldsymbol{S}$ is a binary image of size $M \times N$, with $S(m,n) = 0$ corresponding to the black pixels and $S(m,n) = 1$ for the white pixels. The host image $\boldsymbol{H}$ is of size $P \times Q$. To keep the shape of the secret image unchanged, the aspect ratio of the secret image should be identical to that of the host image (i.e., $P/Q = M/N$). The embedding scheme includes the following two steps.

Step 1. Divide the original host image into blocks of size $P/M \times Q/N$.

Step 2. If $\begin{cases} S(m,n) = 0, H(i,j) \text{ is compressed with Q2,} \\ S(m,n) = 1, H(i,j) \text{ is compressed with Q1,} \end{cases}$ with all $i, j$ satisfy $\lceil i/(P/M) \rceil = m$; $\lceil j/(Q/N) \rceil = n$.

After the embedding process, we save the embedded image in JPEG format, but the quality should be as high as possible, in that way the pixel values will not be quantized and rounded again with another large QTs. Quality for saving the final image after embedding data was chosen 100 or 95 in the experiments for the above reason.

In decoding the recipient first detects the regions with different quality factors to extract the pattern of the message embedded to the host image (the scrambled pattern). As the pattern is extracted, the recipient can rescramble the pattern with the key which was shared in advance with the sender to reconstruct the hidden message.

For example, consider a scheme in which black pixels are embedded in the lower-quality blocks. The process to extract the information bit embedded in the host image is then as follows. The recipient resaves the received JPEG image $I_1$ with the lower quality to yield other version $I_2$ of received image $I_1$; $I_2$ is in lower quality $Q2$. The next step is to subtract $I_1$ to $I_2$ to obtain the difference image. In the difference image, after scaling, the blocks corresponding to the blocks that were compressed with lower quality will appear as black blocks, while other blocks shows as noisy ones and will be decoded as white pixel.

The reason we scale the difference image is that in practice, the difference between the resaved image and the original one is very small, so the difference images would be in very dark

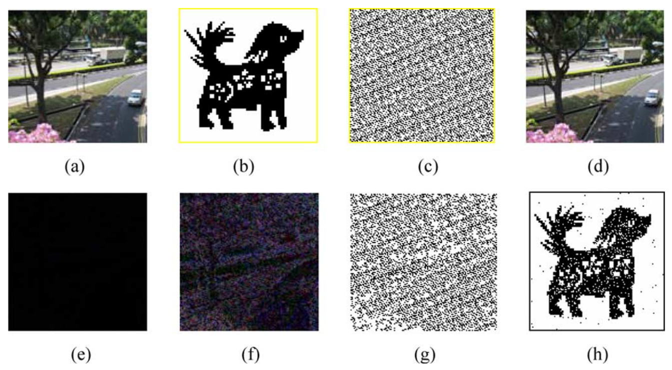
Fig. 3. Example of embedding/extracting processes: (a) Original image, (b) Secret message, (c) Secret message after encryption, (d) Embedded image, (e) Difference image, (f) Difference image after scaling 25 times, (g) Extracted embedded pattern, and (h) Secret message after decoding. $\mathrm{PSNR} = 42.9\ \mathrm{dB}$.

fashion or blur details inside. The embedded regions, after subtraction is assumed to have low value (in fact almost 0), thus after scaling those regions should be still very dark. We propose a simple scheme to extract hidden message based on thresholding as follows:

Step 1. Resave the embedded image with lower quality factors (Q2).

Step 2. Subtract the resaved version to the original embedded image to obtain the difference image.

Step 3. Divide the difference images into blocks of size $8 \times 8$

Step 4. The greatest and least sum values in the difference image are denoted as $s_1$ and $s_2$. Select the difference image which has the greatest difference $(s_1 - s_2)$ as the best candidate. Set the threshold as the average of $s_1$ and $s_2$.

Step 5. If the sum of the pixel value in each block is smaller than the threshold then decode a black pixel, otherwise decode a white pixel.

In Matlab, the function `imwrite` allows us to write JPEG image from a matrix that represents the uncompressed image, with specified quality factor defined by user. The default QTs at different quality factors of other software such as Photoshop were also examined to test the ability to extract the embedded information with blind guess of QTs.

TABLE I
PSNR (DB) WITH DIFFERENT SETS OF Q1 AND Q2

| Q2 \ Q1 | 95 | 90 | 85 | 80 | 75 | 70 | 65 | 60 | 55 |
|---|---|---|---|---|---|---|---|---|---|
| 90 | 44.1 | | | | | | | | |
| 85 | 43.2 | 43.4 | | | | | | | |
| 80 | 42.2 | 41.8 | 43.2 | | | | | | |
| 75 | 41.2 | 41.3 | 41.6 | 43.2 | | | | | |
| 70 | 40.7 | 41.0 | 40.7 | 41.5 | 43.3 | | | | |
| 65 | 40.2 | 40.3 | 40.3 | 40.6 | 41.4 | 43.0 | | | |
| 60 | 39.7 | 39.8 | 40.1 | 39.9 | 40.5 | 41.4 | 43.1 | | |
| 55 | 39.4 | 39.5 | 40.0 | 39.6 | 39.8 | 40.4 | 41.3 | 43.0 | |
| 50 | 39.0 | 39.2 | 39.4 | 39.4 | 39.2 | 39.7 | 40.2 | 41.0 | 42.7 |
| 45 | 38.7 | 38.8 | 38.9 | 39.2 | 38.9 | 39.1 | 39.5 | 39.9 | 40.8 |
| 40 | 38.3 | 38.4 | 38.4 | 39.0 | 38.6 | 38.5 | 38.7 | 39.0 | 39.5 |
| 35 | 37.8 | 37.9 | 38.1 | 38.2 | 38.5 | 38.3 | 38.0 | 38.3 | 38.5 |

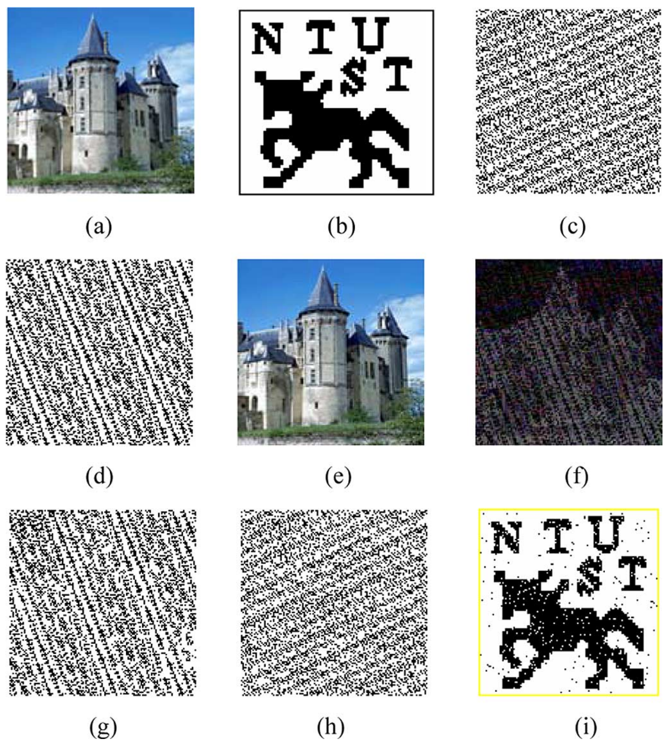
Fig. 4. Two-round encryption result. (a) Original image, (b) Secret message, (c)–(d) Secret message after the first and second encryption round, (e) Embedded image, (f) Difference image after scaling 25 times, (g) Extracted patterns, (h)–(i) Decoded secret message. $\mathrm{PSNR} = 39.8\ \mathrm{dB}$, $\mathrm{CDR} = 94.4\%$.

## III. Experimental Results

Fig. 3 shows an example using our proposed scheme. Fig. 3(a)–(d) associate to the original image of size $1024 \times 1024$, the secret message of size $128 \times 128$, the secret message after the permutation stage and embedded image, respectively. The permutation parameters are $w = 2$ and $n = 83$ (which means to decode, we need to apply the same algorithm with $w = 2$ and $n' = 109$).

The performance of the proposed scheme is assessed by PSNR, obtained by comparison between the host image and the embedded image, and Correct Decoding Rate (CDR) value. CDR is defined by the ratio of the number of pixels decoded correctly and the number of total pixels in the secret image. In Fig. 3, the quality factors are $Q1 = 95$ and $Q2 = 80$. The embedded image looks identical to the host image, with PSNR of 42.25 dB. The remaining subimages show the difference between the embedded image with its resaved version at quality factor 80, and then scaled 25 times. Table I shows the PSNR with different pairs of Q1 and Q2. Notably if the different between Q1 and Q2 is around 10 to 15, the PSNR between the original image and the embedded image is around 40dB. The closer Q1 and Q2 are, the higher the PSNR between the two images is.

Fig. 4 shows another example using the proposed scheme. The host image and the secret message are of sizes $512 \times 512$ and $64 \times 64$, respectively. The quality factors are set as $Q1 = 80$ and $Q2 = 55$. The secret message is encrypted in two rounds with parameters $w_1 = 1$, $n_1 = 40$ and $w_2 = 2$, $n_2 = 66$. After the recipient can apply the permutation algorithm in reverse order, first with $w_2 = 2$ and $n_2' = 30$, then $w_1 = 1$ and

TABLE II
CDR(%) WITH DIFFERENT SETS OF Q1 AND Q2

| Q2 \ Q1 | 100 | 95 | 90 | 85 | 80 | 75 | 70 | 65 | 60 |
|---|---|---|---|---|---|---|---|---|---|
| 95 | 79.8 | | | | | | | | |
| 90 | 85.2 | 78.7 | | | | | | | |
| 85 | 90.1 | 84.5 | 80.6 | | | | | | |
| 80 | 92.7 | 88.6 | 84.1 | 81.2 | | | | | |
| 75 | 95.3 | 91.4 | 89.5 | 84.7 | 83.1 | | | | |
| 70 | 96.1 | 92.9 | 92.4 | 86.9 | 84.4 | 83.0 | | | |
| 65 | 97.2 | 94.8 | 93.9 | 90.6 | 87.3 | 84.3 | 85.4 | | |
| 60 | 98.1 | 96.8 | 95.0 | 93.7 | 89.1 | 88.0 | 88.9 | 87.3 | |

TABLE III
AVERAGE PSNR AND CDR(%) VERSUS DIFFERENCE BETWEEN Q1 AND Q2

| Q1-Q2 | 5 | 10 | 15 | 20 | 25 | 30 | 35 | 40 |
|---|---|---|---|---|---|---|---|---|
| PSNR | 43.3 | 41.6 | 40.6 | 39.6 | 39.3 | 39 | 38.8 | 38.3 |
| CDR | 82.8 | 85.6 | 88.4 | 91.2 | 93.1 | 94.6 | 96.5 | 97.5 |

$n_1' = 8$ to obtain the expected result. Fig. 4(a)–(e) show the original image, the secret message, the encrypted secret message, and the embedded image, respectively. In this case the PSNR of the embedded image is approximately 39.8 dB, and the CDR is 94.4%. Fig. 4(i) shows the decoded result.

The correct extraction rate depends on the QTs. Table II shows the CDR with different pairs of Q1 and Q2. In practice, the chosen difference between Q1 and Q2 should not be too small, the ideal range is about 20 to 40. In extracting the embedded information by applying threshold decision to the difference image, sometimes the block that were previously compressed with Q1 quality yields a very small difference to that compressed with quality Q2 (when Q1–Q2 is small). As a result, the CDR cannot reach 100% in most cases due to the poor contrast in the difference image. Table III shows the average PSNR and CDR versus the difference (Q1–Q2). Choosing Q1 and Q2 is a tradeoff, in which a high CDR is demanded while keeping quality of the embedded image.

In addition, the employed QT can also be considered as a key to guarantee high security. In Fig. 1, both QTs can provide quality factor approximately of 80%, yet the coefficients are apparently differen. One QT is used by the function imwrite in Matlab, the other is used in ACDSee software. The blocks that were used to embed information only appear as black block (minimum difference) in the difference image if the embedded image is resaved at quality Q2 by using the exact QTs for Q2. Otherwise, the difference in the block is not minimized and results in the CDR decreases significantly (reduces more than 20% according to experiments).

To evaluate the capability to evade steganalyser schemes, we tested the output embedded JPEG images with two state-of-the-art steganalysers: stegdetect [9] and Farid's scheme [10]. Both schemes are blind, universal steganalyser. Stegdetect runs statistical tests to determine if steganographic content is present, and also tries to find the system that has been used to embed the hidden information. The systems that stegdetect can easily indentify are jsteg, outguess and jphide. Farid's scheme uses the first four moments of the wavelet coefficients over N subbands. The result of using steghide is impressive: for all the embedded images, with different pairs of Q1 and Q2, steghide returns negative results (no steganographic content). Even with the quality difference of 70, stegdetect classifies the output image into innocent group. For Farid's scheme, the detection rate performance is 4.7% as $(Q1\text{-}Q2) \leq 20$ and increases to 13.3% as (Q1-Q2) is over 60. The experimental results show that the proposed scheme is capable of evading steganalyser schemes with very high chance.

## IV. CONCLUSIONS

This scheme has its beauty because it is very plain, clear, and highly practical. Another advantage is that it does not employ any "traditional" space such as $\mathrm{LSB}(\pm)$ plane or DCT domain. The proposed scheme works only by switching between the two different quality factors, combining with an encryption procedure to enhance the security, thus in general it would be safe under some steganalysis schemes, as shown in the experimental results. The strength of the scheme is based on the strength of the encryption key, and the embedding technique using different QTs is to add a supplemental protective layer for the encrypted message, by hiding it into a normal, innocent JPEG image. Nowadays, most still images are compressed in JPEG format, the simplicity of the proposed scheme would make it a very practical candidate for secret communication.

## REFERENCES

[1] K. Raja and C. Chowdary, "A secure image steganography using LSB, DCT and compression techniques on raw images," in *3rd Int. Conf. Intelligent Sensing and Information Processing*, 2005, pp. 170–176.
[2] Z. Ni and Y. Shi, "Robust lossless image data hiding designed for semi-fragile image authentication," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 18, no. 4, pp. 497–509, 2008.
[3] G. Gul and F. Kurugollu, "A novel universal steganalyser design: "LogSv"," in *IEEE Int. Conf. Image Processing (ICIP 2009)*, Cairo, Egypt, 2009.
[4] H. Farid, "Exposing digital forgeries from JPEG ghosts," *IEEE Trans. Inform. Forensics Security*, vol. 4, no. 1, pp. 154–160, Mar. 2009.
[5] W. B. Pennebaker and J. L. Mitchell, *JPEG: Still Image Data Compression Standard*. New York: Springer, 1993.
[6] [Online]. Available: http://www.impulseadventure.com/photo/jpegquantization.html
[7] G. Voyatzis and I. Pitas, "Applications of toral auto-morphisms in image watermarking," in *Int. Conf. Image Processing, Proceedings*, 1996, vol. 1, pp. 237–240.
[8] H. K. Tso *et al.*, "A lossless secret image sharing method," in *8th Int. Conf. Intelligent Systems Designs and Application*, 2008, pp. 616–619.
[9] [Online]. Available: Detection framework can be found at http://www.citi.umich.edu/u/-provos/papers/detecting.pdf. Supporting technical document and the utility can be downloaded at http://www.outguess.org/
[10] S. Lyu and H. Farid, "Steganalysis using color wavelet statistics and one-class support vector machine," in *SPIE Symp. Electronic Imaging*, 2004 [Online]. Available: http://www.cs.dartmouth.edu/farid/research/