PSTEG: STEGANOGRAPHIC EMBEDDING THROUGH PATCHING

Kyle Petrowski [a], *Mehdi Kharrazi* [a], *Husrev T. Sencar* [b], *Nasir Memon* [b]

[a] Dept. of Electrical and Computer Eng., Polytechnic University, Brooklyn, NY, USA.
[b] Dept. of Comp. and Inf. Science, Polytechnic University, Brooklyn, NY, USA.

ABSTRACT

In this paper we propose a novel approach to image steganography in which embedding is done without making explicit modifications to the image; that is, the embedding distortion introduced to the cover image is both perceptually and statistically ensured to be less detectable. If an image is divided into blocks and each block is hashed, then the hash values could represent the embedded message content. A set of replacement blocks can be obtained by consecutively capturing images of the same scene (or resampling the incident light). Since image content remains the same and noise is sampled, the set of replacements are statistically compatible while still providing unique hash values. With such an approach the embedder can choose the block with hash value corresponding to the message content without violating any of the natural image statistics. When applied to JPEG images, experiments show that this technique can achieve embedding rates of .063 bits per DCT coefficient or .157 bits per nonzero DCT coefficient while still remaining undetectable by Farid's universal steganalysis tool.

1. INTRODUCTION

Steganographic embedding techniques share the common goal of maximizing the embedding rate while minimizing the distortion caused by the embedding process. For example Outguess [3] preserved the global DCT histogram, F5 [4] minimizes embedding distortion via minimizing the number of modifications through employing *matrix embedding*. On the other hand, the technique in [5] models statistical properties of DCT coefficients and preserves the statistics of the perceptually important image component while the embedder operates on the perceptually insignificant component. Perturbed Quantization technique [6] incorporates an innovative coding technique which embeds into select DCT coefficients.

In general, the main weakness with all embedding techniques is that they modify the natural image statistics one way or the other, thus making steganalysis possible by detecting deviations from natural image statistics. But would it be possible to embed a secret message in a cover image, without any modifications to the image. Well it is possible but not practical. For example, let's consider cover images obtained trough a digital camera. Then, if the goal is to embed n bits of information in an image, one could keep capturing images until the LSB of n locations of interest (defined by a secret key) match the message bits. But the number of images one has to capture to obtain an image with the right LSBs, for a reasonable message length, will be large on average and thus such an approach would be impractical.

On the other hand, it is well known that image capturing devices add some noise to the captured image. This noise is due to different components of the image capturing device. Using the above observation, and the fact that consecutive images captured from a constant scene will be different due to the device noise, we propose a new embedding technique called steganographical embedding trough patching or PSteg in short. PSteg creates stego images by patches obtained from multiple captured copies of the selected cover image. Such an approach will reduce the embedding distortion to the image capturing device noise which is less prone to perceptual and statistical detection.

The rest of this paper is organized as follows: in section 2 we first introduce the concept behind PSteg. In Section 3, we cover practical issues, and in 4 we show some experimental results. Finally in section 5 we conclude with discussion and future work.

2. PSTEG

At the heart of PSteg is the availability of multiple copies of a selected cover image. The copies will be very similar although not identical. The block diagram of the embedder is given in Figure 1. Images are broken into blocks of arbitrary size according to a predefined pattern.

For each block coordinate (in the pattern) the unique blocks are identified over all copies of the images. The number of unique blocks indicates the payload (number of bits) that can be carried by the block at the selected block coordinate. For example if there are 8 unique alternates for each image block, then one can embed 3 bits per block.

N. Memon thanks Ton Kalker for suggesting this approach.
This work was supported by AFRL Grant No. F30602-03-C-0091

The embedding is done by breaking up the message bit string into pieces that match the available payload of each block. For this, each image block is hashed to the number of bits to be embedded in that block. The stego image is then patched together from a proper set of blocks whose hashes, when combined, match the embedded message. Decoding is done simply by breaking the stego image into blocks using the predefined block pattern and hashing them. Practical concerns are further addressed in the following sections.

3. PRACTICAL ISSUES

3.1. Image acquisition

A practical implementation of PSteg requires a number of additional considerations. Most image capture devices exhibit some nonlinear characteristics, often induced by heat. This results with a set of images whose content differs in significant ways, perhaps by a DC offset. If left unchecked, such a problem would result in stego images which are easily distinguishable from their cover counterparts, both visually and statistically. To compensate for such effects one of the images could be chosen as a base image and each potential block replacement could be adjusted to match the chosen characteristics of the respective block of the base image. Limiting the normalization process to minimum processing will preserve the uniqueness properties.

In the case of a DC offset, or brightness variation, the normalization could be accomplished by subtracting a block's mean and then adding the mean of the respective base patch, or in the frequency domain, simply using the DC value of the base patch. Since the brightness difference may not be constant across the entire image, the normalization should be performed independently for each block instead of over the entire image set. Another problem arises when there is a global or local shift between the copies of the captured images. Most generally, uncontrolled device movements yields global shifts whereas local shifts are due to changes in the captured scenery.

3.1.1. Flatbed Scanner

Image acquisition was initially attempted with a consumer grade flatbed scanner. The advantages of using a scanner are that it provides a controlled environment for image capture and can be used on any printable photograph. Many scanners also output the data uncompressed and relatively unprocessed. But there are a number of disadvantages such as the brightness variation, which must be normalized. Additionally, the movement of the scanner's mechanical arm seems to result in a subpixel shift between sample images. While there are techniques for eliminating subpixel shift, the effect of such techniques may make the resulting stego image detectable.

3.1.2. Digital camera

The advantages of using a digital camera are that it does not require an internal light source and can depend on a consistent external source. Also, since there is no internal mechanical movement, any relative movement is external to the camera and can be controlled. For example to avoid camera movement, the shutter could be triggered remotely.

Although consumer grade digital cameras tend to be less flexible in giving access to raw, unprocessed data than scanners, they commonly use the ubiquitous JPEG format which may mask the distortion created by PSteg blocks. Disadvantages of using a digital camera are that it also requires brightness normalization.

3.2. Coding

Ideally an infinite number of images (of the same scene) could be taken to guarantee that each message symbol is available. However, the time and memory constraints render such an approach impractical. JPEG compression, conformation to a base image, and hashing collisions can further reduce the availability of message symbols.

In practice, a hashing function should be chosen that minimizes the number of collisions. Some possibilities are taking a subset of the 128 bit MD5 hash, taking a subset of the DJB2 hash, or *xor*ing together all disjoint subsets of the 128 bit MD5 hash of a particular size. Experiments show that *xor*ing MD5 hashes performs best in the context of PSteg.

However, even with 150 JPEG compressed images, when conformed replacement blocks are unique, the *xor*ed MD5 hashes exhibit too many collisions to produce all 128 values which would result in an 7 bit capacity for each block. In order to have all possible hash values available the hash length would have to be reduced to 3 or 4 bits. Instead of accepting such a devastating reduction in information carrying capacity, a simple coding scheme could be used to avoid unavailable hash symbols.

In the coding scheme used, for any particular block we hash all its replacements to one of 128 symbols. However, we know that for any particular block some of the symbols may be missing. We determine an upper bound on the missing symbols, call it M, so that there are at most M symbols missing for any particular block and we pick M symbols to be *null* symbols. Since we have 128 total symbols and M *null* symbols, there remain 128 minus M symbols for message content and we will code our message stream with these 128 minus M remaining symbols.

However, in practice, over saturated and under saturated regions of images have too many missing hash values, making M too large and reducing the embedding rate below acceptable levels. To fix this problem, blocks whose DC values are over-saturated or under-saturated beyond a certain

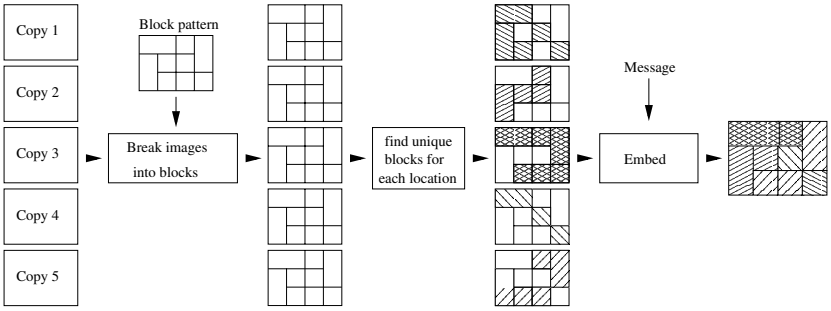

Fig. 1. Block diagram for PSteg embedder.

threshold are ignored. Although not optimal, the following heuristic works quite well:

$Threshold_{high} = DC_{max} - \lceil 10log_2 N_{DC_{max}} \rceil + 20$

$Threshold_{low} = DC_{min} - \lceil 10log_2 N_{DC_{min}} \rceil$

where, N is the number of DC values equal to Max or Min.

Before embedding the message, the *null* symbols should be selected as the first M values in a previously agreed upon sequence of hash values and M should be embedded in the parity of the DC coefficients of the first 7 non-saturated blocks in an agreed upon sequence of blocks.

To simplify the illustration of this technique, assume that we are hashing to the following six, instead of 128, symbols: A, B, C, D, E, and F. We might know that at most two of them may be missing. So we pick two *null* symbols (for example B and E). Now we know we have 6 symbols total and 2 *null* symbols, leaving 4 symbols for message content. We may match them to binary message content like this: 00-A, 01-C, 10-D, 11-F.

Assume we have an image consisting of 4 blocks. Since there are at most two symbols missing from any given block, our available symbols may look something like this:

Block1- A,C,D,E,F	Block2- B,C,D,F
Block3- A,C,D,F	Block4- B,C,D,E

If we have a message stream like this: 1000011110... when converted to our symbol set it would look like this: DACFD... and we would construct our image as follows:

Block1- D	Block2- B
Block3- A	Block4- C

The receiver knows from the hashing function that there are 6 symbols total. The only additional information that the receiver needs is how many *null* symbols are needed for this particular image. Once the receiver knows the number of *null* symbols he can pick them sequentially from an agreed upon permutation of the total available symbols (for example BEAFDC). So we embed the number of *null* symbols in the parity of the DC coef of the first 3 blocks (only once for the entire image). In this case we embed 010 to indicate that the first two symbols in the sequence (B and E) are the *null* symbols. Thus the receiver would decode the PSteg message as follows: DBAC->10_skip_00_01->100001...

4. EXPERIMENTAL RESULTS AND COMPARISON

For image acquisition an "Olympus 300 Digital" was selected. It was set to capture 640x480 color images at its highest quality setting, which uses an automatically adapted custom quantization table for each image. Avoiding the image acquisition problems discussed above, 89 different scenes were captured with 150 sample images for each scene. Before processing, the default images were converted to gray scale, by stripping off the color information.

The block size was taken as the 8x8 JPEG blocks and each block's 63 AC coefficients were hashed with MD5 (the DC coefficient was reserved for brightness adjustments). Each of the 128 bit hash values were further reduced to 7 bit hash values by *xor*ing together all of the 16, 8 bit, disjoint subsets and taking the 7 LSBs of the result.

For each of the 89 scenes acquired 10 stego images were created, each with a different base sample image and a different random message embedded with the coding scheme discussed above. For comparison each stego image's respective base sample image was taken as a cover image. The size of the message embedded in each stego image was recorded along with the number of nonzero DCT coefficients. The average embedding rates for the 890 stego-images were .063 bits/coefficient and .157 bits/nonzero coefficient, with standard deviations of .0067 and .027, respectively.

In order to test the undetectability of PSteg to steganalysis algorithms, we used the universal steganalysis technique proposed by Farid [7]. As for the classifier the freely available LibSvm[8] package was used with a RBF kernel. After being converted from the DCT domain to the spatial domain, 74 statistics were computed for each of the 1780 images via wavelet decomposition. Fifteen of the scenes were chosen at random for testing, while the remaining scenes were used for training. This partitioning was repeated a total

of ten times, with different random subsets used for training and testing each time.

For each of the ten partitionings the SVM was trained with the statistics from the 1480 image training subset (74 different scenes, 10 stego images and 10 cover images for each scene). Afterward the trained classifier was tested against the previously unseen 300 images (150 cover, 150 stego). Over the average of the ten iterations, the SVM classifier was only able to correctly classify **50.4%** of the test images at a false positive rate of **44.1%** and a false negative rate of **55.2%** and with standard deviations of .9%, 14.9%, and 14.7%, respectively.

Before considering how PSteg compares with other embedding techniques, it is important to note that the training/testing procedure outlined in this paper was designed to reveal the robustness/reliability of the proposed embedding algorithm. Rather than attempting to discriminate stego images from randomly selected cover images, the procedure was designed to discriminate directly between PSteg-embedded images and their cover counterparts. Furthermore, multiple resampled versions of the cover images along with multiple versions of the stego images were used to give the classifier as many chances as possible to find the fundamental differences between the cover and stego images of a particular scene. Intuitively, there is no reason why comparing stego images to randomly selected cover images or providing one example of cover and stego for a particular scene would yield better performance results. It is in these two important ways that the procedure for the benchmarks of alternative image steganographic techniques given in [9] differs from the procedure used for PSteg. Additionally, the procedure in [9] used a linear kernel for the SVM classifier, instead of an RBF kernel. Although computational requirements may render the linear kernel more practical, the RBF kernel is certainly more rigorous. Despite these differences in benchmarking methodology, PSteg still has performance results competitive with the best of alternative embedding techniques. One thing worth mentioning is that although the alternative benchmarks were performed at an embedding rate of .2 bits/nzdct, if comparative data for .15 bits/nzdct was available in [9] then it would have been included here instead. There is no reason why PSteg cannot perform at a higher embedding rate with more samples. Approximately .15 bits/nzdct was chosen arbitrarily to demonstrate proof of concept. Comparative results extracted from the ROC curves of [9] are presented in Figure 1.

5. CONCLUSION AND FUTURE WORK

A novel steganographic embedding technique that embeds information without making explicit changes to an image has been demonstrated at moderately high embedding rates of .063 bits/coefficient and .157 bits/nonzero-coefficient, while remaining completely undetectable against Farid's [7] universal steganalysis technique. These results are in conformance with other state-of-the-art embedding techniques.

Comparison of Detection Rates				
Embedding Technique	PSteg	PQ	Outguess	F5
False Positive	.44	.44	.44	.44
False Negative	.55	.56	.46	.45
Total Correct	.505	.50	.55	.555

Table 1. All embedding rates .2 bits/nzdct except PSteg.

However, PSteg still has room for improvements. Characteristics yet to be determined are how much the bits/nonzero-coefficient can be increased by reducing the quality factor and how many samples can be included before reaching the capacity limit. A more sophisticated coding scheme would also probably improve embedding rates and detectability results could be made more rigorous by attempting other methods of steganalysis. Future research could explore a number of extensions, such as applying the technique to color images and other image formats (JPEG2000, bitmap, etc.). Furthermore, a more rigorous analytical investigation of this technique might reveal possible weaknesses which could be used for further improvement.

6. REFERENCES

[1] G.J. Simmons, "The prisoners problem and the subliminal channel," *CRYPTO*, pp. 51–67, 1983.

[2] M. Kharrazi, H. T. Sencar, and N. Memon, "Image steganography: Concepts and practice," *to appear in Lecture Note Series, Institute for Mathematical Sciences, National University of Singapore*, 2004.

[3] N. Provos, "Defending against statistical steganalysis," *10th USENIX Security Symposium*, 2001.

[4] A. Westfeld, "F5a steganographic algorithm: High capacity despite better steganalysis," *4th International Workshop on Information Hiding.*, 2001.

[5] Phil Sallee, "Model-based steganography," *International Workshop on Digital Watermarking, Seoul, Korea.*, 2003.

[6] J. Fridrich, M. Goljan, and D. Soukal, "Perturbed quantization steganography with wet paper codes," *ACM Multimedia Workshop, Magdeburg, Germany, September 20-21*, 2004.

[7] S. Lyu and H. Farid, "Detecting hidden messages using higher-order statistics and support vector machines," *5th International Workshop on Information Hiding.*, 2002.

[8] Chih-Chung Chang and Chih-Jen Lin, *LIBSVM: a library for support vector machines*, 2001, Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

[9] M. Kharrazi, H.T. Sencar, and N. Memon, "Benchmarking steganographic and steganalysis techniques," *SPIE Symposium on Electronic Imaging, San Jose, CA*, 2005.