A Cropping Resistant Algorithm for Identification of the Author of Photographs

**Abstract**

In this paper we present an algorithm which can be used for identification of the author of the photographs. Most modern smart phones have fingerprint sensors, as the cameraman clicks the image, his fingerprint will be embedded in the image. This would be very useful for getting a hard evidence for copyright claims on images. The algorithm can also be used by web crawlers searching for identifying unauthorized use of intellectual property. Our algorithm is robust enough to work even if the copyright violator crops out a part of the image.

**Keywords** – Steganography; image cropping; cropping attack; copyright protection; smart phone; smart phone camera; image security; fingerprinting; stochastic; biometric embedding;

**Introduction**

The literature of steganography is vast and many authors have presented many ingenious methods to preserve secrecy. However, to the best of our knowledge, no author has assumed the scenario that the attacker will crop the image. All algorithms in the surveyed literature fail under this scenario.

JSteg was one of the first algorithms for JPEG steganography. It stores the hidden data in the LSBs of the DCT coefficients. Initially a sequential and later a pseudorandom embedding path were used [1]. This technique is easily detectable using a histogram attack [2]. This method fails because, when the attacker crops the image, the recompression would overwrite not only the data, but also change the positions of the data.

Permutation straddling of Quantized DCT coefficients has been used for making embedded data hard to detect. The scholars in [6] have used a Genetic Algorithmic approach to find a permutation which minimizes the footprint and that makes it less detectable. However, in our scenario the attacker would recompress the image and the data would get lost. In fact cropping the image by four pixels may cause a “spacial resonance” and it can be a good way to detect if a similar technique was used [4,5].

Steganography on cropped images has been done in [3], however the cropping is not done by the attacker. The image is cropped into several non-overlapping images and they need to be merged before the extraction process.

Several scholars have embedded the hidden information by Discrete Wavelet Transform [8,9]. However when the image is cropped and recompressed, all the stored data becomes irrecoverable. In [8], the bounding box for cropping is considered as the steganography key. In [9], the cropping is methodical and not done by the attacker.

**Methodology**

The objective is to embed the biometric onto the carrier image. In order to decrease the size of the hidden data, only two most significant bits of the gray scale biometric are considered. Single bit depth can also be used. The pixels from the biometric are taken in raster format and then they are put in the 2nd and 3rd least significant bits of the carrier image. The 1st least significant bit will be used for storing guidelines because changes in the 1st LSB later are the least detectable. The position of the pixel whose LSB is to be replaced is determined by a pseudo random generator. Different functions like pseudo random generators and Lagrange polynomials [3] have found their use for this purpose.

In other words, the pseudo random number generator (PRG) generates the x and y coordinates in the carrier image. The seed of the PRG serves as the key. Since PRGs generate the same sequence every time for the same seed, it can be used to recover the positions of the hidden data.

**Protection against cropping**

The first problem that arises when the attacker crops the image is that certain parts of the biometric image are lost. Ideally every possible cropped image should have sufficient data to reconstruct the carrier image. This problem is solved by having redundancy of data and this is achieved by running multiple passes on the carrier image.

For random numbers taken from uniform distribution, doubling the number of passes also doubles the expectation of reconstructing the complete image. Another benefit of using this technique is that even if all the pixels are not recoverable, it does not fail completely. Rather a complete albeit noisy image is obtained. Median blur can be used to remove the salt and pepper noise if needed.

In a very rare occasion, the PRG may generate the same x, y pair twice. A majority voting system is used to counter this rare event. This also has the added benefit that if a portion of the image is retouched, the majority voting should be able to right the wrong.

Another problem that arises is when the image is cropped the x, y pairs are offset by the starting coordinates of the cropped image. Unless the offset is figured out, the PRG would not do much good. This problem is tackled using guidelines.

In order to avoid noise or interference, the LSBs of the carrier image is set to 0. Vertical lines are drawn through every pixel whose horizontal coordinate is a perfect square and horizontal lines are drawn through every pixel whose vertical coordinate is a perfect square. The carrier image is flipped and the process is repeated. The result is demonstrated in Fig 1.

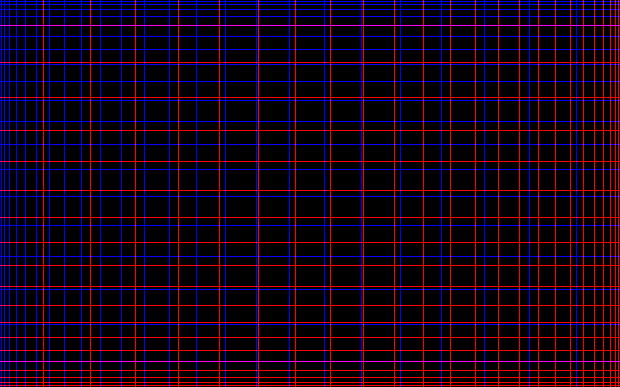


Figure 1: Illustration of the guidelines in the LSB of the image

When the image is cropped, the distance between two vertical or horizontal lines can be found by counting intermediate pixels. Since, the position of lines follows the equation: , the separation of lines is . Say, if the separation of the two lines in the cropped image is 16 pixels, i.e. . Substituting the value of in we get, . So, the position of the line in the carrier image was 64 pixels from whichever side we are currently operating.

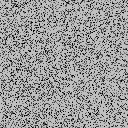


Figure 2: If the cropped image is smaller than two guidelines, the carrier dimensions and subsequently the embedded image cannot be recovered.

If the position of the line in the cropped image is 30 pixels, then the position of the 0th pixel of the cropped image is pixels in the carrier image. Now we can offset the output of the PRG by and reconstruct the hidden image. We have to repeat the process to get both x and y offsets.

The upper limit of randomly generated values is also needed by the PRG to generate the same sequence. This upper limit is the dimensions of the carrier image. Even if the dimensions of the carrier image are unknown, it can be calculated by finding the x and y offsets of the bottom right corner of the screen using the same process as described above.

* (i)

**Robustness and Practicality**

It is easy to see that if the calculated offset or the dimensions of the image is wrong by even one pixel, the hidden data would be irrecoverable. In order to find the offset for any side, we need at least two lines.





Figure 3: (8 passes) Larger the size of the cropped image, more embedded data is recovered

Let y be the pixel with biggest indices in the longest side which is also a perfect square. Using the formula, , the position of the line just before that would be at .

* (ii)

If the dimensions of the carrier image are already known, all four corner points are not needed. It is sufficient to have any two adjacent corner points. When the image is crossed by midway, the other side can now be used to calculate the offset. So, effectively the dimensions of the image are halved. The equation now becomes,

* (iii)

Since we are storing one pixel of the hidden image in one pixel of the carrier image, the size of the biometric should be at most this size if we expect to recover it perfectly in the worst case scenario.



Figure 4: (32 passes) As number of passes are increased, more pixels can be recovered.

Let h be the number of pixels in the hidden image and c be the number of pixels in the carrier image. Therefore the expectation of recovering a pixel of hidden image from n pixels of the carrier image is,

* (iv)

Expectation of recovering entire hidden image

* (v)

After s passes, the expectation becomes

* (vi)

We can see if n is the minimum number of pixels in a cropped image that the guidelines can support, then s must be chosen in a manner such that .

* (vii)

**Resistance to Attacks**

The easiest way to detect if this algorithm has been used on an image is to check the 1st bit plane. Also the easiest way to destroy the recoverability of the hidden is to wash the 1st bit plane and crop the image by 1 pixel from each side. Using Linear Feedback Shift Register instead of naïve guidelines might fix the detectability problem.

**JPEG and Lossy Compression**

JPEG uses Discrete Cosine Transform to remove most high frequency data from the image, because it is not easily visible to the human eye. Unfortunately, the proposed method relies heavily on storing the hidden data in the high frequency domain. However, JPEG does not remove all the high frequency data. So, if the data is embedded and the carrier image is recompressed multiple times then each time a little bit of high frequency data is left behind. After sufficient passes, the hidden image can be recovered with a good enough visual fidelity.



Figure 5: (32 passes) extraction process on un cropped JPEG file

This approach completely fails when the image is cropped. The guidelines as discussed above rely on the fact that the bit plane they are embedded on is completely clean. Since bit planes get contaminated with bleeding effects (Fig 6.) with JPEG, they become completely useless. Even with the use of LFSR (Linear Feedback Shift Register) as an alternative to the guidelines, the bleeding effects (as seen in Fig 7.) still make it impractical.



Fig. 6: Blurry effects on the guidelines due to JPEG compression (only X axis shown for clarity)



Fig 7: Bleeding effects and distortions due to JPEG compression on a simple LFSR

**Storing Textual Data**

Since, all data in computers are encoded in binary, a string of text can be converted into an image. ASCII uses 8 bits to represent a character. Bits in pairs of two are taken and used as the MSB of each pixel in the secret image. The string is then repeated for additional redundancy. The recovering process extracts the MSBs and assembles the bit pairs into ASCII.

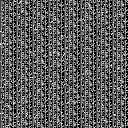
 

Fig. 8: Left, the text “The quick brown fox jumps over the lazy dog.” Encoded as an image. Right, the same image upon recovery

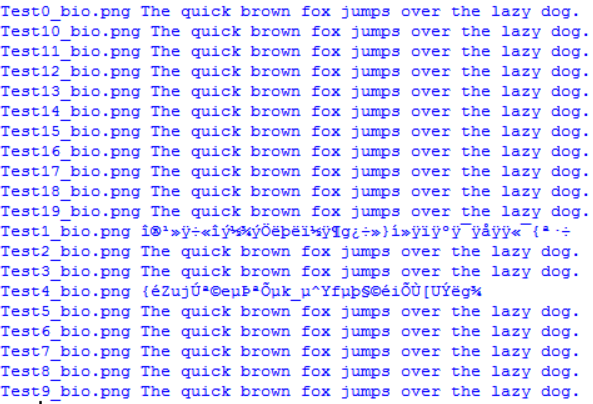


Fig 9. Conversion of the recovered image back to text

**PSNR study**

For JPEG the compression and recompression causes loss in image quality. Each time the cycle completes, the PSNR of the carrier is traded for the PSNR of the extracted image. This is shown in Fig. 10.



Fig. 10. X axis – Number of recompressions. Y – axis PSNR. Blue line indicates the PSNR of the carrier image while the red line is the PSNR of the biometric.

In case of PNG the PSNR of the carrier image does not vary with the number of passes and hence has been omitted from the graph. Its PSNR value is easily over one hundred. The PSNR of the extracted image increases monotonically with increasing number of passes as shown in Fig 11.

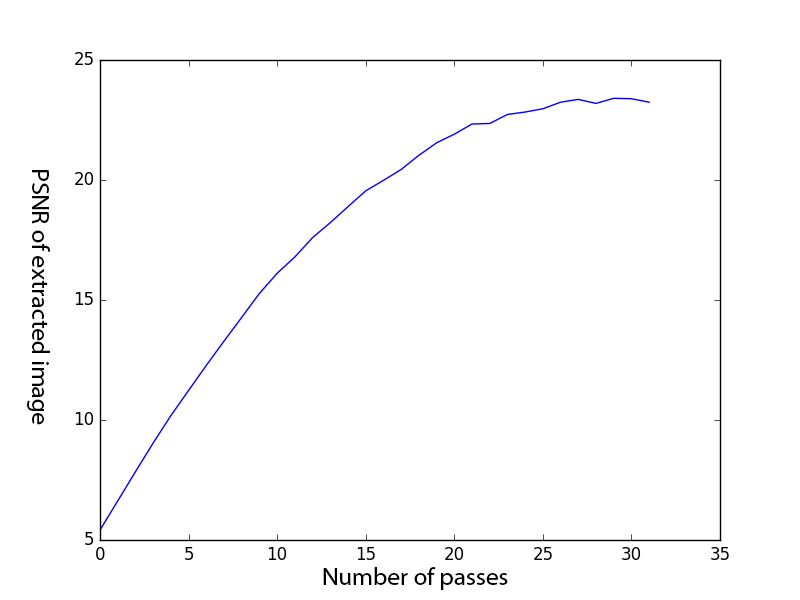


Fig. 11. X-Axis is the number of passes. Y-axis is the PSNR of the extracted image.

**Conclusion and Future Work**

In this paper we have dealt with the problem of cropping attacks on the carrier media. We have also explored the pitfalls when dealing with JPEG and its lossy compression. The technique could have been extended to JPEG compression only if a reliable method of finding the offset coordinates was found. The source code is available for download at (???). The procedure for reproduction is also bundled along.

The problem of having redundancy is the low embedding capacity. Salience maps [7] can be used to find regions of interest and then we can embed the secret data in that region only. This might improve image wide PNSR as we do not have to rely so much on random redundancy.

**References**

1. Fridrich, J., Pevný, T., & Kodovský, J. (2007, September). Statistically undetectable jpeg steganography: dead ends challenges, and opportunities. In Proceedings of the 9th workshop on Multimedia & security (pp. 3-14). ACM.
2. Westfeld, A., & Pfitzmann, A. (1999, September). Attacks on steganographic systems. In International workshop on information hiding (pp. 61-76). Springer Berlin Heidelberg.
3. Potdar, V. M., Han, S., & Chang, E. (2005, August). Fingerprinted secret sharing steganography for robustness against image cropping attacks. In INDIN'05. 2005 3rd IEEE International Conference on Industrial Informatics, 2005. (pp. 717-724). IEEE.
4. Fridrich, J. (2004, May). Feature-based steganalysis for JPEG images and its implications for future design of steganographic schemes. In International Workshop on Information Hiding (pp. 67-81). Springer Berlin Heidelberg.
5. Fridrich, J., Goljan, M., & Hogea, D. (2003, June). New methodology for breaking steganographic techniques for JPEGs. In *Electronic Imaging 2003*(pp. 143-155). International Society for Optics and Photonics.
6. Yu, L., Zhao, Y., Ni, R., & Zhu, Z. (2009). PM1 steganography in JPEG images using genetic algorithm. *Soft Computing*, *13*(4), 393-400.
7. Achanta, R., Hemami, S., Estrada, F., & Susstrunk, S. (2009, June). Frequency-tuned salient region detection. In Computer vision and pattern recognition, 2009. cvpr 2009. ieee conference on (pp. 1597-1604). IEEE.
8. Shejul, A. A., & Kulkarni, U. L. (2010, February). A DWT based approach for steganography using biometrics. In *Data Storage and Data Engineering (DSDE), 2010 International Conference on* (pp. 39-43). IEEE.
9. Kapoor, K. (2016, February) Digital Image Steganography based Security Model using wavelet for Internet voting Environment. International Journal of Advanced Research in Electronics and Communication Engineering (IJARECE) Volume 5, Issue 2