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. The technique we used for protection against cropping however does not work for JPEG images unless the bounding box of the cropped region is externally provided.

**Keywords** – Steganography, cropping attack

**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 literature we have surveyed fails 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 [8,9,10,11]. However when the image is cropped and recompressed all the stored data becomes irrecoverable. [10] has taken the bounding box for cropping as the steganographic key. In [11], the cropping is methodical and not done by the attacker.

**Methodology**

We have two images, the carrier or cover image and the biometric that we wish to embed. To decrease the size of the hidden data, we extract only the two most significant bits of the grey scale biometric. Single bit depth can also be used. We take the pixels from the biometric as raster and put them in the 2nd and 3rd least significant bits of the cover image. The 1st least significant bit will be used shortly for a different purpose. 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 cover image. The seed of the PRG serves as the key. Since PRGs generate the same sequence every time for the same seed, we can use it to recover the positions of the hidden data.

**Protection against cropping**

The first obvious 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 original image. This problem is solved by having redundancy. Multiple passes are made on the image. Since the PRG we have used generates random numbers from uniform distribution, doubling the number of passes, doubles the chances of reconstructing the complete image. Another benefit of using this technique is that even if all the pixels are not recoverable, we do not get an incomplete image. Rather we get the complete albeit noisy image. 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. We have implemented a majority voting system 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 when we crop the image is that the x, y pairs are offset by the starting coordinates of the cropped image. Unless we figure out how much is being offset, the PRG would not do much good. We have tackled this problems is by introducing guide lines.

We start the process by setting the LSBs of all pixels to zero. Then we draw vertical and horizontal lines with increasing separation. We have drawn a vertical line through every pixel whose horizontal coordinate is a perfect square. Therefore we now have lines at 1, 4, 9, 16,… from the left side of the image. We then rotate the image by ninety degrees and apply it again. We repeat for each side of the image as shown in Fig 1.

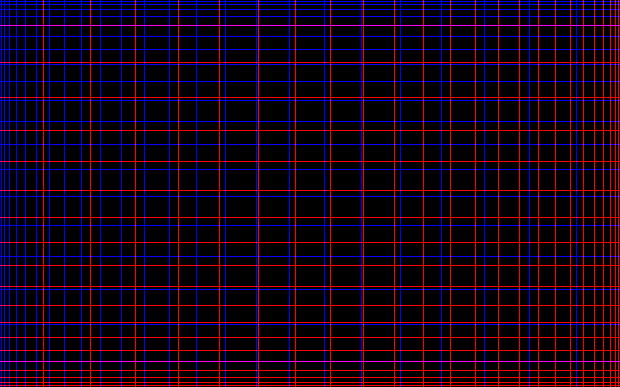


Figure 1: Illustration of the guidelines in the LSB of the image

Say now the image is cropped, we can find the distance between two vertical or horizontal lines. Since, the position of lines follows the equation: , using calculus we find that the separation of lines is . Say, if the separation of the two lines is 16 pixels, i.e. . Substituting the value of in we get, . So, the position of the line in the original image was 64 pixels from whichever side we are currently operating.

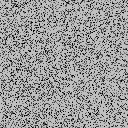


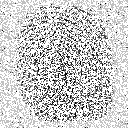
Figure 2: If the cropped image is smaller than two guidelines, the original dimensions and subsequently the embedded image cannot be recovered.

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 original 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.

Yet another piece of information the PRG requires is the upper limit of values it can generate. For our purpose it would be the dimensions of the original image. Even if the dimensions of the original 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.

**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.



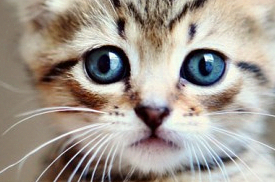


Figure 3: (8 passes) Larger the size of the cropped image, more embedded data is recovered

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 .

If the dimensions of the original image are already known, we don’t need all four corner points. Any adjacent two of them would be sufficient. When we cross the image midway, the other side can be used to calculate the offset. So, effectively the dimensions of the image are halved. The equation now becomes,

Since we are storing one pixel of the hidden image in one pixel of the cover 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 cover image. Therefore the probability of recovering a pixel of hidden image from n pixels of the cover image is,

Probability of recovering entire hidden image

After s passes, the probability becomes

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 .

**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 LFSR 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 method we are using relies heavily on storing the hidden data in the high frequency domain. However, JPEG does not remove all the high frequency data. We take advantage of this fact and run the process of embedding and recompressing several times. Each pass, a little bit of information is retained. 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. We have also explored LFSR (Linear Feedback Shift Register) as an alternative to the guidelines but 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, we can convert a string of text into an image. In our case we do that by taking two bits at a time and setting them as the MSBs of the image to be hidden. ASCII uses 8 bits to represent a character, so we need 4 pixels to represent a character. The string is then repeated for additional redundancy. The recovering process is exactly the same but we have to again take the pair of bits from pixels and assemble them into text.

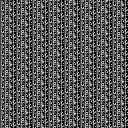
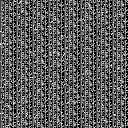
 

Fig. 8: Left, the text “The quick brown fox jumps over the lazy dog.” Encoded as an image. Right, the same image upon recovery

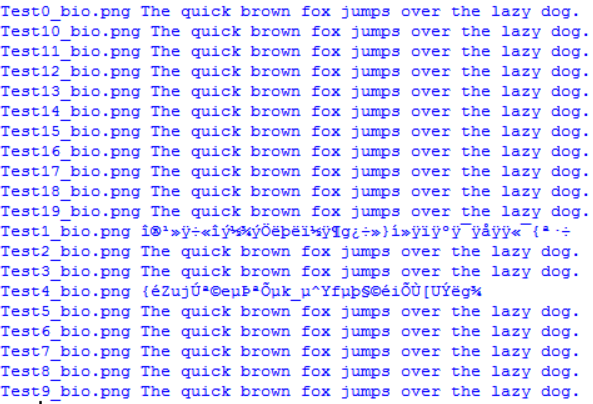


Fig 9. Conversion of the recovered image back to text

**Conclusion and Future Work**

In this paper we have dealt with the problem of cropping attacks on the cover 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 our data in that region only. This not only improves image wide PNSR but also increases embedding capacity 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. Kude, M. M., Borse, M., & Harshvardhan, R. K. A DWT Based Steganography Using Haar Transforms.
9. Swetha, V., & Kumari, V. S. R. INTERNATIONAL JOURNAL OF ENGINEERING SCIENCES & RESEARCH TECHNOLOGY SKIN TONE BASED IMAGE STEGANOGRAPHY USING PARTICLE SWARM OPTIMIZATION.
10. 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.
11. Kapoor, K. 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, February 2016