A Steganographic Algorithm for Identification of the Author of Photographs from Cropped Stolen Photos

Author (AU)

Address (ADR)

Abstract: In this paper, we present a blind steganographic algorithm which can be used for identification of the author of the photographs which were copied without permission. Most modern smartphones 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. Since it a blind steganographic algorithm, it does not need anything other than the cropped image to reproduce the hidden data.

Keywords: Steganography; image cropping; cropping attack; copyright protection; smartphone; smartphone camera; image security; fingerprinting; stochastic; biometric embedding;

Reference: Author(s) (2017), A Steganographic Algorithm for Identification of the Author of Photographs from Cropped Stolen Photos, *Int. J. Computational Science and Engineering*, Vol. X, No. Y4, pp.000–000.

Biographical notes: (ABS)

1 Introduction

As more people get access to the internet all around the world, the threats of intellectual property theft get increased. Professional photographers spend a significant amount of time and skill to take photographs. Not to mention that the gear they use is rather expensive.

There are many images on the internet which are so unprotected that they can be simply copied from the website and used for commercial purposes or otherwise without giving proper credit or compensation to the photographer. The photographer has no way of knowing where his photo is being used as he, by himself, cannot check all websites on the internet.

A few firms provide machine learning based solutions which crawl popular websites and screen for similar looking photographs. However, it picks up all photographs which look similar. For example, it would pick up all photographs of Eiffel tower taken from the same angle and time of the day. The photographer is now presented with the difficult task of knowing which one is his from hundreds of false positives.

Steganographic solutions can be used to tackle the problem with false positives. The crawler can try to extract some data like say a hash from the image. If the hashes match then it is a perfect match and the photographer can file for copyright violation.

The hash also provides a hard evidence as the defender cannot say that it was one another picture taken from the same angle.

The proposed algorithm is designed while keeping in mind that when the image is clicked, the fingerprint of the photographer is immediately embedded into the image. This fingerprint can then be used as evidence. If the author is not interested in embedding his fingerprint, he can also opt to embed any other string like a hash in the photograph. This is also supported.

Another reason for choosing fingerprints is that it is very robust in terms of missing data. Even if all pixels of the fingerprint could not be recovered, the human brain can still quite reliably compare and match two fingerprints. Convolutional auto-encoders can be used to automate this process. Wang et al (2014) achieved a 93.1% accuracy using stacked sparse auto-encoders. Computers greatly enhance the scalability of the operation. The web crawlers can automatically detect and verify if the fingerprints match. The technique proposed by Tang et al (2016) can detect a fingerprint in 0.45 seconds on average on Graphics Processing Unit. It uses a fully convolutional network to learn convolutions instead of using manually crafted convolutions.

Hashes and encrypted texts on the other hand are quite vulnerable to one bit off errors. Both the human brain and neural networks can fill in the blanks in the case of a missing pixel or two. The robustness of neural networks is great for solving problems of this class.

The algorithm functions even when the image is cropped down to a reasonable size. In case a portion of the image is retouched, the algorithm is strong enough to handle it. If that does not work, the retouched part of the image can be cropped out and the algorithm can be run on an untouched portion of the image.

2 Literature Survey

The literature of steganography is vast and many authors have presented many ingenious methods to preserve secrecy. However, very few scholars have worked on this particular problem statement.

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 was used. Westfield et al (1999) demonstrated that this technique is easily detectable using a histogram attack. In our case, 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. Yu et al (2009) have used a Genetic Algorithmic approach to finding 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, Fridrich et al (2003,2004) demonstrated that cropping the image by four pixels may cause a "spatial resonance" and it can be a good way to detect if a similar technique was used.

Potdar et al (2005) have done steganography on cropped images, 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. Al-Afandy et al (2016) have designated predefined areas for cropping. Singh, S., & Siddiqui, T. J. (2016) and Wamiliana et al (2015) do not recover the coordinates of the cropped image from the stego-image itself.

Several scholars (Kapoor, K. 2016; Shejul, A. A., & Kulkarni 2010) have embedded the hidden information by Discrete Wavelet Transform. Vargese J et al (2016) have used DFT and Singular Value Decomposition (SVD) to hide the data in different frequency bands. Shejul, A. A., & Kulkarni (2010) used the bounding box for cropping as the steganography key. In the literature of Atee et al (2015), Deshmukh, P. R., & Gawande, S. V. (2016), Kapoor, K. (2016) and Kong et all (2015) the cropping is methodical and not done by the attacker. Jiang, N., & Wang, L. (2015) have used quantum steganography based on Moire Pattern. They have explored a line removal cropping attack where non-adjacent parallel lines are removed.

PourArian et all (2016) tackle a similar problem statement. The scholars have used DWT for the embedding. For redundancy, they have embedded the data once in the red channel and once in the blue channel but before in the blue channel, they have rotated the channel by 180 degrees. Position identification is done by a marker pixel group. The identification algorithm is discussed in detail in section. For encryption, they have used Arnold Cat Map. Vladimir Arnold demonstrated its effects in the 1960s using an image of a cat, hence the name.

3 Methodology

The objective is to embed the biometric onto the carrier image. The smaller the object to be hidden is, the easier it is to hide it. Fingerprints do not require very high graphical fidelity. So, 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 color channels are alternated. The 1st least significant bit will be used for storing guidelines because changes in the 1st LSB are the least detectable. The process of storing the guidelines would be explained in detail in~~a~~ section 4. The position of the pixel whose LSB is to be replaced is determined by a pseudo-random generator. Potdar et al (2005) have used Lagrange Polynomials for this process.

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 by extracting all the pixels residing in the positions given by the PRG. In the case of a cropped image, the given coordinate could be outside the carrier image. In that case, it is safely ignored.

4 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 biometric image. This problem is solved by having redundancy of data and this is achieved by running multiple passes on the carrier image. The redundancy should be sufficient to have all the information to reconstruct the hidden image in any cropped part of the photo while not being so excessive that the quality of the carrier image is compromised.

For random numbers taken from the uniform distribution, doubling the number of passes also doubles the expectation of reconstructing the complete image. This process has its limits and is graphed in Figure 11. Another benefit of using this technique is that even if all the pixels are not recoverable, it does not fail completely. Rather a

complete but 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. However, it is not mandatory to have this system. Storing the votes takes a considerable amount of memory. Also, counting the votes takes time. Assuming insertion and lookup take constant time, the memory and time complexity is in the order of Θ(nk) where n is the number of pixels of the biometric and k is the number of passes. The process of extraction has the exact same time complexity but takes longer as it has to count all the votes. There is a total of n * k votes i.e. each n pixels have k votes.

The sequence that the PRG would generate is relative to the origin of the original, un-tampered image. As soon as the image is cropped, the origin of the cropped image differs from the origin of the original image. This means that every point generated by the PRG would have to be translated by what the coordinates of the cropped image were in the original image. As an example, let an image be cropped from (100, 100) to (200, 200). So, now the (0, 0) pixel in the cropped image is actually the (100,100) in the original image. The PRG, by itself, has no way of knowing that. This offset must be supplied externally, (-100,-100) in this case. If the PRG generates (150,160) for the original image, the same pixel would now have the coordinates (50, 60) in the cropped image. The process of finding this offset automatically and efficiently without having the original image at hand is the most challenging aspect of this problem.

Therefore, some additional data has to be bundled with the embedded image such that the extractor is able to automatically figure out the starting coordinates of the cropped image. This is done by generating 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. Since this operation is performed on the 1st LSB of the carrier image, it is imperceptible. So, in order to illustrate the process, the MSB of the image was modified as shown in Figure 1.

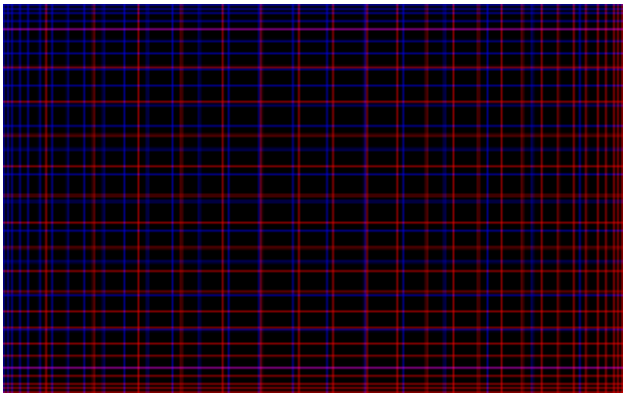


Figure 1: Illustration of the guidelines which are used to find the offset of the cropped image.

When the image is cropped, the distance between two vertical or horizontal lines can be, on the other hand, by counting intermediate pixels. Since, the position of lines follows the equation: $y = x^2$, the separation of lines is $\frac{dy}{dx} = 2x$. Say, if the separation of the two lines in the cropped image is 16 pixels, i.e. $2x = 16 \Rightarrow x = 8$. Substituting the value of x in $y = x^2$ we get, $y = 64$. So, the position of the line in the carrier image was 64 pixels from whichever side we are currently operating.

After finding the actual offset of the line, we find the offset of the origin. In order to do that, the position of the line in the cropped image is subtracted from the position of the line in the original image. Continuing the previous example, 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 64 – 30 = 34 pixels in the carrier or original image. This gives the distance from the margin. In order to find the bounding box, all the four margins must be calculated. This process is repeated four times to find the bounding box of the cropped image i.e. top, bottom, left and right.

The PRG sequence only needs the starting positions to be offset by. However, uniform random number generators also need an upper bound for the randomly generated numbers. In this case, this upper bound would be the dimensions of the original image. Even in the case where dimensions of the original image are not known, the algorithm would still function. The extractor can calculate the dimension of the original image as shown in equation (i).

$$\begin{aligned} dimensions\ of\ original\ image \\ &= upper\ left\ offset \\ &+ dimensions\ of\ cropped\ image \\ &+ lower\ right\ offset \quad [Eq.(i)] \end{aligned}$$

5 Robustness and Practicality

One of the shortcomings of this algorithm is that even if the offsets are wrong by one pixel, the data becomes completely irrecoverable. This would happen if there is any misreading or tampering of the guidelines. Fortunately, any

two guidelines can be used to find the offset. The offset can be calculated from all the pairs and then a voting can take place. So, even if a few of them are tampered or corrupted, the voting should provide a reasonable margin of error.

The minimum dimension of the cropped image required is that many pixels which have enough data to find the offset of the cropped photo. Since only two un-tampered guidelines are required to find the offset from that side, the minimum dimension required is the maximum separation of two guidelines in the worst case scenario. Due to the choice of the guideline generating equation, the separations of lines increase quadratically from one end to the other. Naturally, the last and second last lines in a direction have the largest separation. This is the smallest image that can be cropped with the guarantee that the offset can be recovered. Smaller crops can also have sufficient data but there is no guarantee.

The concept is expressed mathematically as follows. Let y be the pixel with biggest indices in the longest side which is also a perfect square. Using the formula, $y = x^2$, the position of the line just before that would be at $\left(\sqrt{y} - 1\right)^2$.

$$The\ minimum\ dimension\ of\ cropped\ image = y - \left(\sqrt{y} - 1\right)^2 [Eq.(ii)]$$

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,

$$Minimum\ dimension\ of\ cropped\ image = \sqrt{\frac{y}{2}^2} - \left(\sqrt{\frac{y}{2}} - 1\right)^2 \text{ [Eq. (iii)]}$$

Since we are storing one pixel of the biometric 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. However, more than one pixel can be embedded in every pixel of the carrier image. This can be done by running the algorithm more than once on different bit levels. This would inevitably lead to a loss of image quality.

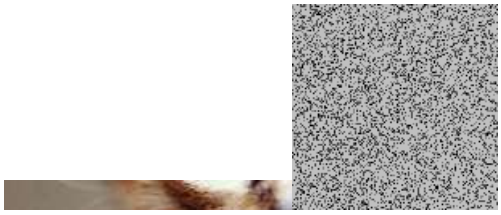


Figure 2. Illustration of the failure state when the dimension of cropped image is lesser than equation (ii).

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

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

So far the number of passes being chosen for the experiments was arbitrary. Anyone implementing the algorithm might need to know the exact number of passes according to his project requirement.

Let us take a scenario where there are c empty boxes, and h unique balls to hide. The balls are distributed uniformly. Then some boxes are discarded which may or may not have balls in them. Say, n boxes are left over after the discarding process. When n = c, we can recover all the balls. When n = 0, nothing can be recovered. Using this analogy, the boxes are the pixels in the carrier image, the biometric pixels are the balls and n is the number of pixels in the cropped image.

Let h be the number of pixels in the hidden image, n be the number of pixels in the cropped image and c be the number of pixels in the original image. Then the number of pixels that can be recovered is,

$$Number\ of\ recoverable\ pixels = \frac{hn}{c} \text{ [Eq. (iv)]}$$

Unless with perfect luck, all the h balls were in the n boxes which weren't discarded, the number of balls that can be recovered would always be lesser than the total number of balls. As mentioned earlier, the h balls are unique. If we want to recover more unique balls, we need to have redundancy. If we have s copies of each unique ball, such that $hs <= c$, then we can hope to find more balls.

On multiplying s with equation (iv) we get

$$Number\ of\ recoverable\ pixels\ after\ s\ passes = \frac{hsn}{c} \quad [Eq.(v)]$$

In the perfect case scenario, we expect to find all h unique pixels of the biometric in a cropped image of size n. Equating h with equation (v) we get,

$$s = \frac{c}{n}\ [\text{Eq. (vi)}]$$

It might be unrealistic to expect to recover all h pixels from the cropped image. So a k term is added which is the fraction of the biometric which is practical to recover.

$$s = \frac{kc}{n}\ [\text{Eq. (vii)}]$$

Figure 11 demonstrates a monotonically increasing relation between the number of passes and the PSNR of the recovered image. Since k cannot be greater than one, the effect of diminishing returns of PSNR with increasing passes is also documented.

The fluctuations observed near the tip of the line in figure 11 are due to overwriting of an old pixel with a new one. This happens when PRG generates the same coordinate more than once. This conforms to the fact that as the number of passes increase, the probability of PRG coordinate collisions also increases. This can be witnessed in the figure from the decrease in the smoothness of the line with increasing number of passes. It is the job of the majority voting system to diminish this effect but if there are too many collisions, the majority can vote for the wrong pixel.

6 Resistance to Attacks

This algorithm was designed to be used by civilians at the commercial level. It does not fare very well against the methods used by professional analysts and steganographic experts. It is possible to encrypt the image before the embedding process using existing reliable and popular methods. Chunking can be used to localize damages. Having a very high level of redundancy also works quite well. In section 8, we have explored the storage of textual data and obtained quite satisfactory results.

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. The guidelines are clearly the weakest point of the proposed algorithm in case of an attack.

Linear Shift Feedback Registers or LSFR (Murase, M. 1992) generates a sequence of bits deterministically. An useful property of LSFR is that given a sequence, $s_i, s_{\{i+1\}}, \dots, s_{\{i+m-1\}}$, it is possible to uniquely identify i by trying all possible values. The mapping from i to $s_i, s_{\{i+1\}}, \dots, s_{\{i+m-1\}}$ is bijective. Where m is a large integer and 2^m is significantly larger than the dimensions of the image.

LSFR is not only an alternative but it might actually be a better solution to the problems discussed above because to the naked eye it is indistinguishable from noise. Extracting the position using LSFR is slower than the guidelines approach. The value of m also becomes an issue. We have found that despite of its problems, the guidelines approach is more practical.

PourArian et all (2016) proposed the usage of a marker pixel group for locating the center of the image, which is a white pixel surrounded by four black pixels at the center of the image. The problem with this approach is twofold. Firstly, there is a possibility that the image might have the same white pixel surrounded by four black pixels by coincidence. Secondly, the disadvantage is that if the attacker crops within a quadrant, the marker would not help.

The algorithm is also robust to other geometric attacks like rotation and scaling. In the case of rotation, the guidelines can be used to reorient the image. In the case of scaling, the guidelines can give the scaling factor and all the coordinates generated by the PRG can, with it.

7 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 iterations, the hidden image can be recovered with a good enough visual fidelity.

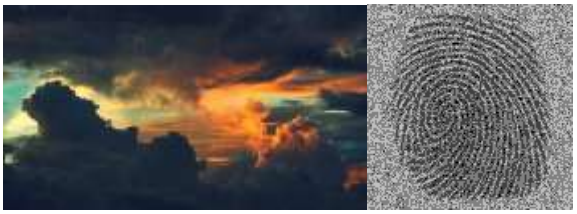


Figure 5: (32 passes) Extraction process on not-cropped JPEG file

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, as shown in Figure 7, they become completely useless. Due to the periodic nature of the DCT, some extra lines are inserted at regular intervals as shown in Figure 6. Also, the lines get fuzzier this makes the calculation of the offset difficult.

LFSR does not survive the JPEG compression either. Having each bit of the LSFR occupy one pixel is out of the question due to the lossy nature of JPEG. So, each bit is represented by a block of pixels with the hope that its frequency would be low enough to not get removed by the JPEG compression. In Figure 7 a checkerboard pattern is drawn for the simplicity of demonstration. As humans, we are easily able to deduce the pattern from the image on the right but finding the exact coordinates becomes impossible. Also, if we observe closely, the image on the left is affected by the pattern although the pattern was embedded in the LSB before the compression.

Image processing methods like erosion can be used to solve the fuzziness problem with guidelines. The majority voting system can implicitly deal with the problem of extra lines being drawn on the image. It should be noted that in Figure 6, the guidelines are drawn on a blank black image. When this experiment is repeated on a real photo then the contents of the image bleeds into the guidelines depth. This happens to such an extent that the majority voting system cannot handle it. The bleed back effect as seen in Figure 7 left is absent for the guidelines approach possibly due to the fact that the lines are very thin.

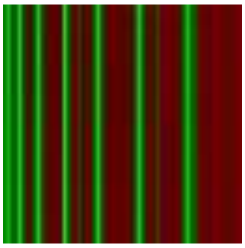


Fig. 6: Blurry effects on the guidelines due to JPEG compression (only X axis shown for clarity)

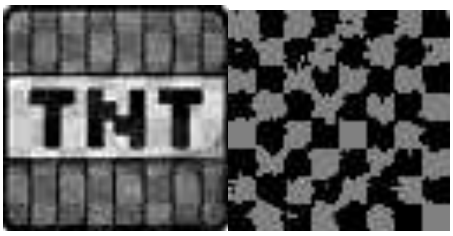


Fig 7: Bleeding effects and distortions due to JPEG compression on a simple LFSR sequence

8 Storing Textual Data

Steganography is often used to store textual data. In this section, we demonstrate that the proposed algorithm has the capability of doing the same. The user can either put the message in plaintext or encrypt it for added security. Since our algorithm is prone to having a few bits off and most encryption algorithms do not work even if a bit is wrong, it is advised to encrypt the text in chunks. This would minimize the fallout.

In practice, the results are not so pessimistic. The only reason that Test 1 and Test 4 failed in figure 9 is because the cropped image was smaller than two guidelines. This meant that the coordinates could not be recovered and therefore the ASCII interpretation was garbage. Textual data is often much smaller than an image, we can tile the text repeatedly for a huge amount of redundancy. The majority voting system then sorts out the discrepancies.

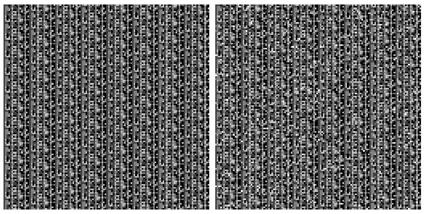
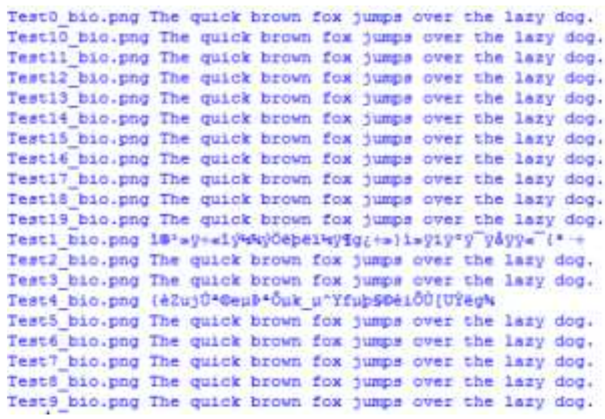


Fig. 8: Left, the text “The quick brown fox jumps over the lazy dog.” Encoded as an image. Right, the same image upon recovery

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.



```
Test0_bio.png The quick brown fox jumps over the lazy dog.
Test10_bio.png The quick brown fox jumps over the lazy dog.
Test11_bio.png The quick brown fox jumps over the lazy dog.
Test12_bio.png The quick brown fox jumps over the lazy dog.
Test13_bio.png The quick brown fox jumps over the lazy dog.
Test14_bio.png The quick brown fox jumps over the lazy dog.
Test15_bio.png The quick brown fox jumps over the lazy dog.
Test16_bio.png The quick brown fox jumps over the lazy dog.
Test17_bio.png The quick brown fox jumps over the lazy dog.
Test18_bio.png The quick brown fox jumps over the lazy dog.
Test19_bio.png The quick brown fox jumps over the lazy dog.
Test1_bio.png [illegible]
Test2_bio.png The quick brown fox jumps over the lazy dog.
Test3_bio.png The quick brown fox jumps over the lazy dog.
Test4_bio.png [illegible]
Test5_bio.png The quick brown fox jumps over the lazy dog.
Test6_bio.png The quick brown fox jumps over the lazy dog.
Test7_bio.png The quick brown fox jumps over the lazy dog.
Test8_bio.png The quick brown fox jumps over the lazy dog.
Test9_bio.png The quick brown fox jumps over the lazy dog.
```

Fig 9. Conversion of the recovered image back to text

9 PSNR study

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

Both the lines in the graph seem to converge with increasing number of recompressions. This observation points to the fact that the process has diminishing returns.

For a practical implementation, the gradient of both curves can be calculated. When the gradient falls below a certain threshold, the process is stopped. Another approach would be to repeat the process only until the carrier image drops to a certain arbitrarily chosen PSNR level or reaches a fixed number of iterations.

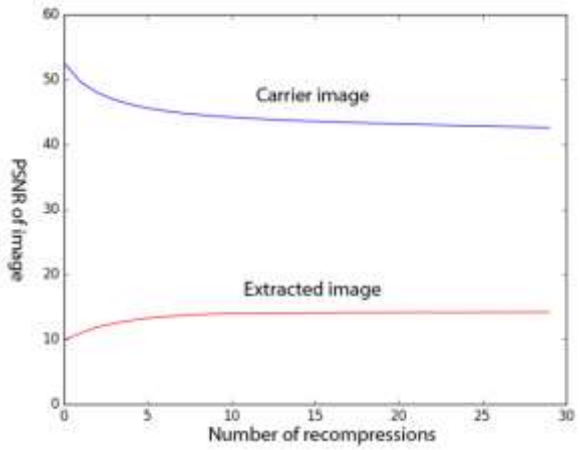


Figure 10: (32 passes) The PSNR of both the carrier image and the extracted image flattens out and converges with multiple iterations.

In the case of PNG, the PSNR of the carrier image does not vary with the number of passes and hence has been omitted from Figure 11. Its PSNR value is easily over one hundred. The image was cropped to 50% of its size.

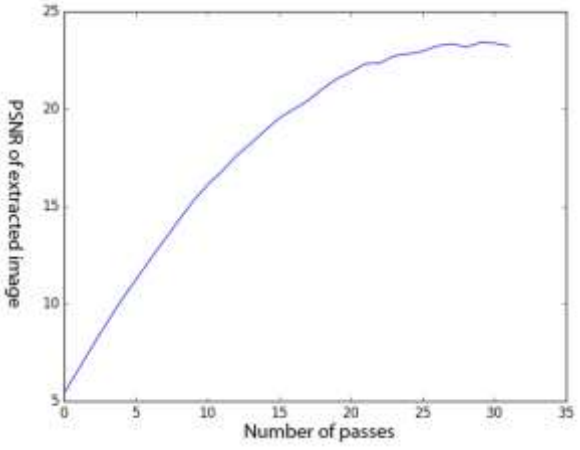


Figure 11: The PSNR of the extracted image increases monotonically with increasing number of passes.

The effects of diminishing returns are also visible in Figure 11. The extracted image from the PNG image has about twice the PSNR of the one extracted from the JPEG image for the same number of passes. This is also considering the fact that the PNG image was cropped while the JPEG image wasn't. This means that the extractor might not have had access to all the pixels of the embedded image while the JPEG image did so.

10 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 Github (URL_REDACTED). The procedure for reproduction is also bundled along.

The problem of having redundancy is the low embedding capacity. Salience maps (Achanta et al 2007) 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.

11 References

1. 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.

2. Al-Afandy, K. A., Faragallah, O. S., Elmhalawy, A., El-Rabaie, E. S. M., & El-Banby, G. M. (2016, October). High-security data hiding using image cropping and LSB least significant bit steganography. In Information Science and Technology (CiSt), 2016 4th IEEE International Colloquium on (pp. 400-404). IEEE.

3. Atee, H. A., Ahmad, R., & Noor, N. M. (2015). Cryptography and Image Steganography Using Dynamic Encryption on LSB and Color Image-Based Data Hiding. Middle-East Journal of Scientific Research, 23(7), 1450-1460.

4. Deshmukh, P. R., & Gawande, S. V. (2016). Secret Image Sharing By Diverse Image Media. International Research Journal of Engineering and Technology (IRJET) Volume: 03 Issue: 05

5. 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.

6. 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.

7. 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.

8. Jiang, N., & Wang, L. (2015). A novel strategy for quantum image steganography based on Moiré Pattern. International Journal of Theoretical Physics, 54(3), 1021-1032.

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

10. Kong, X., Wang, B., Yang, M., & Feng, Y. (2016). Multiple Heterogeneous JPEG Image Hierarchical Forensic. In Advanced Multimedia and Ubiquitous Engineering (pp. 509-516). Springer Singapore.

11. Murase, M. (1992). U.S. Patent No. 5,090,035. Washington, DC: U.S. Patent and Trademark Office.

12. PourArian, M. R., & Hanani, A. (2016). Blind Steganography in Color Images by Double Wavelet Transform and Improved Arnold Transform. Indonesian Journal of Electrical Engineering and Computer Science, 3(3), 586-600.

13. 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.

14. 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.

15. Singh, S., & Siddiqui, T. J. (2016). Copyright Protection for Digital Images using Singular Value Decomposition and Integer Wavelet Transform. International Journal of Computer Network & Information Security, 8(4).

16. Tang, Y., Gao, F., & Feng, J. (2016). Latent fingerprint minutia extraction using fully convolutional network. arXiv preprint arXiv:1609.09850.

17. Varghese, J., Subash, S., Hussain, O. B., Nallaperumal, K., Saady, M. R., & Khan, M. S. (2016). An improved digital image watermarking scheme using the discrete Fourier transform and singular value decomposition. Turkish Journal of Electrical Engineering & Computer Sciences, 24(5), 3432-3447.

18. Wamiliana, W., Usman, M., Azram, M., Elfaki, F. A. M., Hijriani, A., & Panditatwa, P. (2015). Adaptive minimum error least significant bit replacement method for steganography using JPG/JPEG and PNG files. Science International Lahore, 27(6), 4987-4991.

19. Wang, R., Han, C., Wu, Y., & Guo, T. (2014). Fingerprint Classification Based on Depth Neural Network. arXiv preprint arXiv:1409.5188.

20. Westfeld, A., & Pfitzmann, A. (1999, September). Attacks on steganographic systems. In International workshop on information hiding (pp. 61-76). Springer Berlin Heidelberg.

21. Yu, L., Zhao, Y., Ni, R., & Zhu, Z. (2009). PM1 steganography in JPEG images using genetic algorithm. Soft Computing, 13(4), 393-400.