Encrypted Semi-Reversible Data Hiding Technique Based on CUDA-C

Ahmed Abdo, Marcus Chow, and Kiran Ranganath

Abstract— **This paper extends the block-based semi-reversible data hiding technique to colored images using CUDA-C. Our implementation is compatible with colored images in BMP file format. This new technique increases the capacity of an image by three times, only with a slight reduction in the Peak Signal to Noise Ratio. We have also added a level of encryption where a key is used to determine the pattern in which the data is encoded and decoded.**

Keywords: Data hiding, block processing, Encryption, histogram shifting.

I. INTRODUCTION

Data hiding is referred to as a process, which is associated with a group of techniques used to secure any data within a host media (like images) with little or no deterioration in the host and the means to retrieve the secure data back from the host [1]. Many image data hiding algorithms have been proposed in the past years. In most cases, the cover media will experience some permanent distortion due to data hiding and cannot be inverted back to the original media even after the hidden data have been extracted out. This kind of data hiding techniques is only applicable to some applications, in which the small distortion due to the data embedding is tolerable. However, in many other special fields, such as medical diagnosis, law enforcement and military imaging, the possibility of recovering the exact original image after the hidden data are retrieved is a desirable property.

In this paper, first a comprehensive survey is performed to introduce the main concepts and main methods people proposed for data hiding and authentication. Then the paper mainly focuses on the implementation of an image watermarking and authentication application on CUDA-C platform. A block-based semi-reversible data hiding algorithm is used for watermarking, and a Peak signal-to-noise ratio between original and retrieved watermark is used to verify the integrity of the received data.

The rest of the paper is organized as follows: experimental results on several images and comparison to the state of the art are presented in Section 4. Finally, conclusions and future research directions are provided in Section 5.

II. BACKGROUND SURVEY

Many image data hiding algorithms have been proposed in the past years. In this section,we will highlight on some of the important ones. Nosrati et al. [1] introduced a method that embeds the secret message in RGB 24 bit color image. This is achieved by applying the concept of the linked list data structures to link the secret messages in the images. First, the secret message that is to be transmitted is embedded in the LSB of 24 bit RGB color space. Next, like the linked list where each node is placed randomly in the memory and every node points to every other node in list, the secret message bytes are embedded in the color image erratically and randomly and every message contains a link or a pointer to the address of the next message in the list. Also, a few bytes of the address of the first secret message are used as the stego-key to authenticate the message. Using this technique makes the retrieval and the detection of the secret message in the image difficult for the attacker.

Kuo et al. [2] [3] [4] [5] presented a reversible technique that is based on the block division to conceal the data in the image. In this approach the cover image is divided into several equal blocks and then the histogram is generated for each of these blocks. Maximum and minimum points are computed for these histograms so that the embedding space can be generated to hide the data at the same time increasing the embedding capacity of the image. A one bit change is used to record the change of the minimum points.

Das et al. have listed different techniques to hide data [6] [7]. The authors have mainly focused on how steganography can be used and combined with cryptography to hide sensitive data. In this approach they have explained and listed various methods like Plaintext Steganography, Still Imagery Steganography, Audio/Video Steganography and IP Datagram Steganography which can be used to hide data. The authors have also elucidated the Steganalysis process which is used to detect if steganography is used for data hiding.

Dey et al. [8] have proposed a novel approach to hide data in stego-images which is an improvement over the Fibonacci decomposition method. In this method the authors have exploited Prime Numbers to hide data in the images. The main agenda is to increase the number of bit planes of the image so that not only the LSB planes but even the higher bit planes can be used to hide to data. This is done by converting the original bit planes to some other binary number system using prime numbers as the weighted function so that the number of bits to represent each pixel increases which in turn can be used hide data in higher bit planes. The authors have also performed a comparison of the Fibonacci decomposition method with the traditional LSB data hiding technique showing that the former outperforms the latter method and comparing Fibonacci Decomposition method with the proposed method which outclasses the former method. Also, the proposed method generates the stegoimage which is virtually indistinguishable from the original image.

Nikolaidis [9] proposed a method that is semi-reversible, in the sense that the hidden data are always extracted accurately. The proposed method does not use any overflow location map, since overflows are systematically avoided. Additionally, although the method works in the spatial domain and is based on histogram shifts and splits, no auxiliary information about histogram peaks and/or zeros is required to be sent along with the message to be hidden. In this method, the image is first divided into non-overlapping blocks of size (K x K). For each block, its histogram h is computed and certain features of it are extracted. Specifically, if h(a) is the first non-zero point of the histogram, that is, if a is the lowest gray level that appears in the image block, and h(b) is the last non-zero point, that is, if b is the highest gray level that appears in the image block, then each image block may fall into one of certain classes, depending on whether we are in the embedding or the extraction phase.

III. Proposed Algorithm

Since Nikolaidis' method is only applicable on gray images. We proposed a method that works on colored images. Moreover, we add a key with the embedded information for a security purpose. We also embed our hiding data using GPU for a faster operation since we use now three colors for embedding. First, the image is first separated into the three colors red, green, and blue. Each color then, is divided into non-overlapping (k x k). A histogram h is applied to each block. h(a) which is the first non-zero point of the histogram, and h(b) which is the last non-zero point will be calculated so that each image block will be falling into one of seven classes. These classes will decide whether they are in eligible for embedding or extraction.

A. *Data Embedding*

In the embedding process, the image block may appear as one of the following classes[9]:

TABLE I
Embedding Classes

Classes	description
Class 1	a = 0 and 255-b > 2
Class 2	a > 2 and 255-b = 0
Class 3	a >1 and 255-b > 1
Class 4	a = 0 and 0 < 255 - b $\geq$ 2
Class 5	0 > a $\geq$ 2 and 255-b = 0
Class 6	a $\neq$ 0 and 255-b $\neq$ 0
Class 7	a = 0 and 255-b = 0

The classes 1,2, and 3 are the ones that can be used for embedding and we call them embeddable. The classes 4,5,6, and 7 are defined as non-embeddable classes and they are not suitable for embedding. The embeddable classes are used because they are appropriate for shifting the histogram with the aim of making a room so that a message can be embedded. Then, the message bits will be embedded next to the pixels that have the maximum value by moving that pixels lift or right by one.

First, we separate the image into the three colors red, green, and blue. Then, we apply a key to the message that will be embedded later. For each color, we will divide it to (k x k) blocks, then, we will check for the embeddable blocks to be used for the embedding the desired message. We will embed the same message to each color for redundancy and robustness purpose.

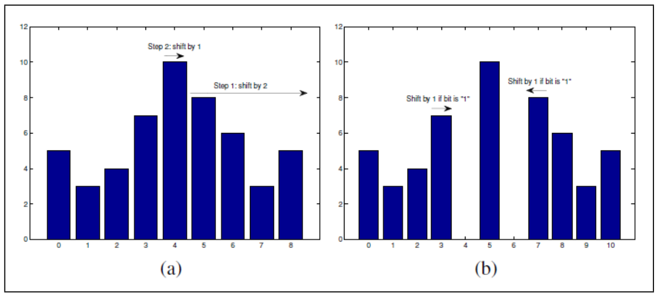

Fig. 1. Class 1 embedding: (a) steps 1 and 2, (b)step 3.

For the class 1 blocks, the histogram part to the right of the maximum bin is shifted right by two. After that, the maximum bin is shifted to the right by one. This is shown in Figure 1(a) and Figure 1(b), where the maximum bin value is number 4. All the bins that are greater than 4 will be shifted right by two. The maximum bin will be shifted right by one. The message bits will be embedded to the left and to the right of the maximum bin.

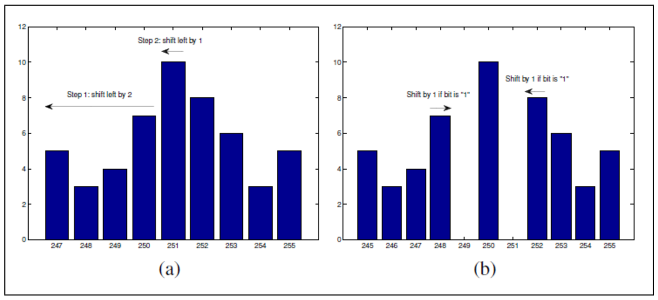

Fig. 2. Class 2 embedding: (a) steps 1 and 2, (b)step 3.

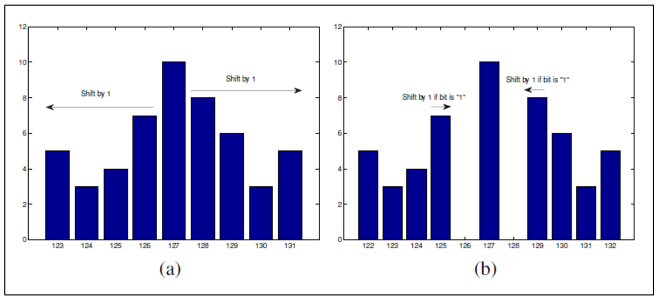

Fig. 3. Class 3 embedding: (a) steps 1 (b) step 2.

During scanning the message bits after applying the key for security purpose, if a 0 bit is encountered, the corresponding bin will not change. If a 1 bit is encountered, then the corresponding bin that are smaller than the maximum bin will be added and the ones that are greater than the maximum bin will be subtracted by one. For class 2 blocks, we will move all the histogram bins that are to the left of the

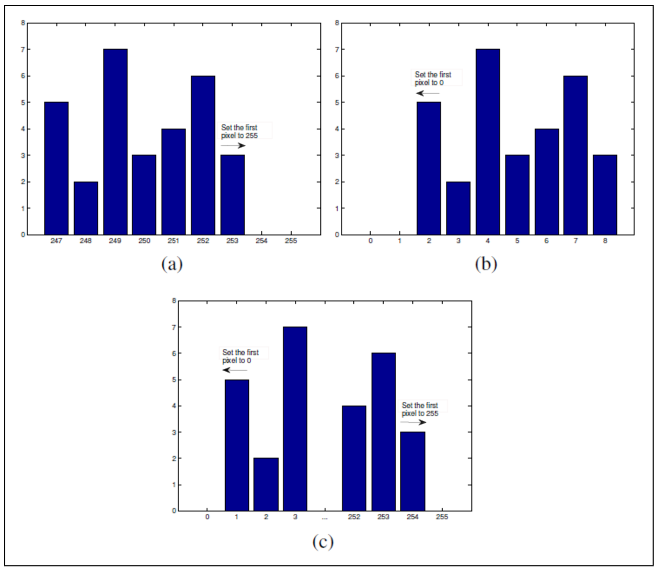

Fig. 4. Non-embeddable classes: (a) class 4, (b) class 5, (c) class 6

maximum value leftwards by two. Subsequently, we shift the maximum bin to the left by one so that a free space will be available by the right and left of the maximum bin. This is shown in Figure 2(a) and Figure 2(b), where the maximum bin is located at 251. After shifting, two rooms are made at bins 249 and 251 so that the color that has a value of 248 will be added by one and a value of 252 will be subtracted by one if the message bit to be embedded is 1. If the block is class 3, the left histogram bins to the maximum bin will be shifted leftwards by one and the right histogram bins to the maximum bin will be shifted rightwards by one, as shown in Figure 3(a) and Figure 3(b), where the maximum bin is the value 127. In the following phase, color values of 125 will be added by one and values of 129 will be subtracted by one if the message bit to be embedded is 1.

Since we change the blocks of classes 1, 2 and 3 to be blocks of classes 4, 5 and 6, we need to change the original blocks of classes 4, 5 and 6 to differentiate between them so that we recognize the blocks where embedding has occurred when we do the extraction process. Therefore, we will to change the blocks of class 4, 5 and 6 blocks to class 7 blocks. For class 4 blocks, we will move the last non-zero histogram bin encountered at the rightmost to 255, as shown in Figure 4(a), where the first color value located in the block that has the highest value will be changed to 255. For class 5 blocks, we move the leftmost non-zero histogram bin to 0, as shown in Figure 4(b). We can see that the first color value that has the minimum value in the block becomes 0. Finally, for class 6 blocks, we move both the greatest values to 255 and the smallest values to 0, as shown in Figure 4(c). As a result, all blocks of classes 4, 5 and 6 become class 7 blocks. We can see the impact of applying the secured hidden message according to the proposed algorithm on the original Lena picture as shown in Figure 5.

B. Data Extraction

During the extraction process, we can find each of the following block classes[9]:

TABLE II
EXTRACTION CLASSES

Classes	description
Class A	a = 0 and 255-b > 2
Class B	a > 0 and 255-b = 0
Class C	a > 0 and 255-b > 0
Class D	a = 0 and 255-b = 2

From the previous table, we can see that class A blocks are those that caused after embedding class 1 blocks, class B blocks are generated from embedding class 2 blocks, and class C blocks are resulting from embedding class 3 blocks. We can also see that class D blocks are coming from the unchanged class 7 blocks or the manipulated class 4, 5 or 6 blocks. Consequently, class A, B and C blocks are known to be the extractable blocks and class D blocks is defined as the non-extractable.

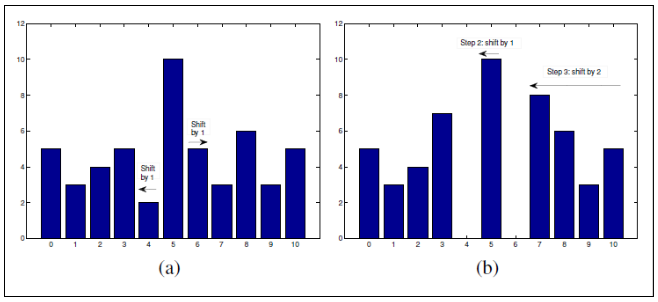

Fig. 5. Class A extraction: (a) steps 1 (b)steps 2 and 3.

The whole purpose for data extraction is to get the secured message bits from each color, use the key to know the original message, and return class A, B and C blocks to their original state. For all these classes, we first find the height histogram bin since we know that it was not changed during embedding. Then, we check the neighbor histogram bins to the maximum one. These bins represent the values that were added or, subtracted by one to hide a message bit 1. On the other hand, the bins that are second to the left and second to the right of the maximum bin are the values that were not altered because a bit 0 was found during the scanning. The raster scan order of blocks and the scan order within each block during the extraction process is the same as the embedding process, so that the message bits are extracted in the correct order. After extracting the message bits in the case of class A blocks, the bins are located to the left and to the right of the maximum histogram bin will be shifted by one to the left and to the right, respectively, as shown in Figure 5(a). Then, the maximum bin will be shifted to the left by one. At the end, all the bins that are placed to the right of the maximum bin will be moved leftwards by two, as shown in Figure 5(a).

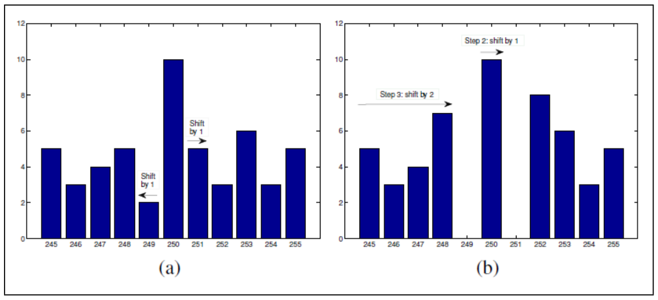

Fig. 6. Class B extraction: (a) steps 1 (b)steps 2 and 3.

For class B blocks, the left and the right bins to the maximum histogram bin will be shifted after extracting the message bits by one to the left and to the right, respectively. The second phase is to move the maximum bin to the right by one. Then, all bins that are to the left of the maximum bin will be shifted rightwards by two, as shown in Figure 6(a) and 6(b).

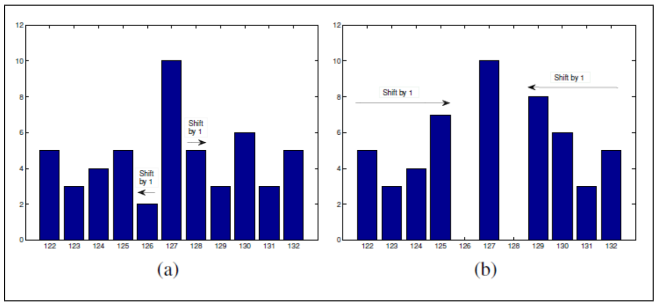

Fig. 7. Class C extraction: (a) steps 1 (b)step.

Finally, for class C blocks, first step is the same as for classes A and B which includes shifting the left and the right bins to the maximum histogram bin after extracting the message bits by one to the left and to the right, respectively. Then, all bins that are located to the left of the maximum histogram bin will be shifted right by one and all bins that are located to the right of the maximum bin will be shifted left by one, as shown in the Figure 7.

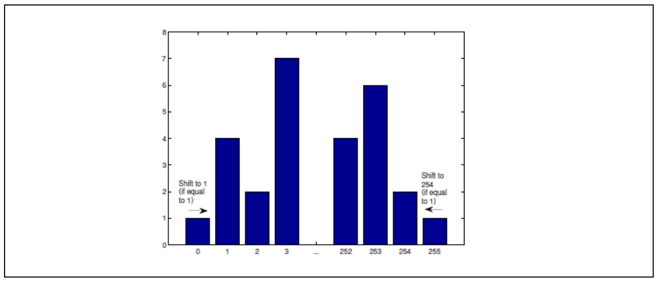

Fig. 8. Non-extractable class

As far for class D blocks which is considered as non-extractable, they will be transformed back to their original shape. For example, the first histogram bin will be shifted rightwards to the next non-zero bin, if the first bin has a value of one. Similarly, the last histogram bin is shifted leftwards to the previous non-zero bin, if the last bin has a value of one. This guarantees that all original class 4, 5 and 6 blocks will be restored, but certain class 7 blocks that were not originally changed might now be distorted. The previous process is shown in Figure 8.

C. *Key Encryption*

In this data-hiding technique, the order in which bits are encoded in a block must be known for the data to be properly decoded. This information allows us to add a layer of encryption into the technique itself. Traditionally, a block will be scanned left-to-right, top-to-bottom, to find pixels that can be encoded. In our technique we use a key to determine the ordering of this scanning operation. This insures that only someone with the key will be able to properly extract the data. In our situation, the sender and receiver both agree to a key before transmitting the image. This way no extra data needs to be sent along with the image.

IV. EXPERIMENTAL RESULTS

To test our technique we encoded a string of 750 characters into images from the USC-SIPI image database. All of these images [10] where of size 512 x 512, bitmap image file type, and in color with a bit depth of 24. All of our tests were run with a block size of 8. As this block size [9] was found to have the highest capacity (Bits per Pixel) and highest pixel to noise ratio (PSNR).

TABLE III
BITS PER PIXEL

Image	Color	Gray
Lena	0.35	0.13
Girl	0.30	0.09
Splash	0.60	0.18
Lake	0.28	0.10
Airplane	0.53	0.18

Our first test was to compare the capacity between color and gray scale images. Table 3 shows our results. As expected color images are, on average, able to store 3 times the amount of bits as gray images. This is because a color image as three separate channels (Red, Green, and Blue) and gray scale images only have one channel (gray) to encode data.

TABLE IV
PSNR

Image	Color	Gray
Lena	48.03	49.1
Girl	37.19	48.8
Splash	46.97	49.5
Lake	42.87	49.1
Airplane	48.90	49.5

Table 4 shows the results from our second test. We compare the Signal to Noise Ration between the original and

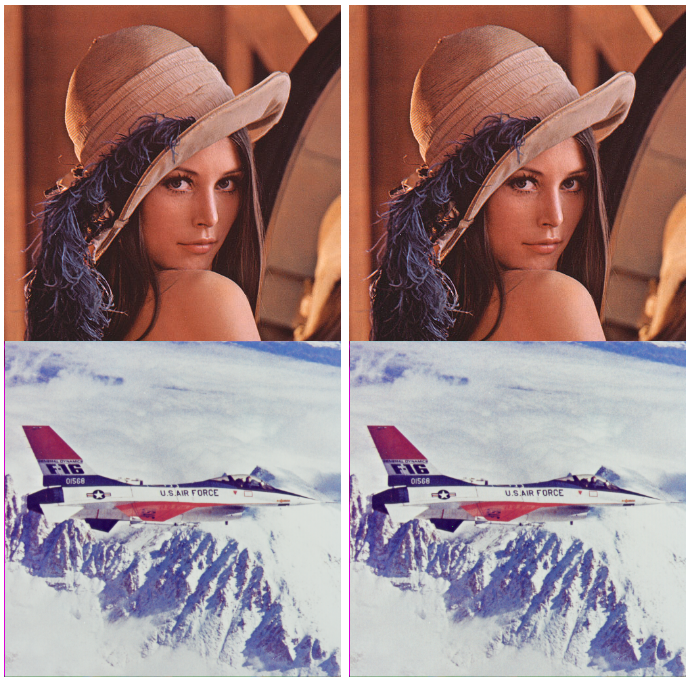

Fig. 9. Left: Before encoding Right: After encoding

embedding images. Embedding colored images results in a lower PSNR value compared to a gray scale image. Figure 9 compares images before and after the encoding process. We believe this has to do with how pixels in non-embeddable blocks are shifted. During the encoding process the high and low value of non-embeddable blocks are shifted to 255 and zero respectfully. This process is not noticeable in gray scale images, but in color images, shifting channel to 255 will dramatically change the color of the whole pixel. This process also produces some "artifacts" where in a block is whiten out to the point where it is visually noticeable. We also do not restore the image, like the previous technique, because our concern is with the data being hidden and not the image itself.

V. CONCLUSIONS AND FUTURE WORK

We have demonstrated that the semi-reversible block-based data hiding technique introduced in [9] can be extended to colored images in the RGB format. When applied to colored images we are able to embed three times the number of bits compared to gray scale images, but with a reduction in the PSNR. The difference in PSNR values widely vary. Sometimes to an extreme where "artifacts" of the embedding process are clearly noticeable. We believe this can be avoided by changing the class definitions and shifting method to better fit a colored image.

Our method also added a level of encryption to the embedding process. Because this technique requires the order in which pixels are scanned to be the same during encoding and decoding, we are able to add a key to the process which will also encrypt the data. By using a key to decide the order in which data is embedded, we are able to insure that no one will be able to decode the data unless they have the key.

REFERENCES

[1] M. Nosrati, R. Karimi, H. Nosrati, and A. Nosrati, "Embedding stego-text in cover images using linked list concepts and LSB technique", Journal of American Science, Vol. 7, No. 6, 2011, pp. 97-100..

[2] Wen-Chung Kuo, Dong-Jin Jiang, Yu-Chih Huang, "A Reversible Data Hiding Scheme Based on Block Division", Congress on Image and Signal Processing, Vol. 1, 27-30 May 2008, pp. 365-369.

[3] Yih-Chuan Lin, Tzung-Shian Li, Yao-Tang Chang, Chuen-Ching Wang, Wen-Tzu Chen, "A Subsampling and Interpolation Technique for Reversible Histogram Shift Data Hiding", Image and Signal Processing, Lecture Notes in Computer Science, Vol. 6134, 2010, Publisher: Springer Berlin/Heidelberg, pp. 384-393.

[4] Chyuan-Huei Thomas Yang, Chun-Hao Hsu, "A High Quality Reversible Data Hiding Method Using Interpolation Technique," IEEE Fifth International Conference on Information Assurance and Security, Vol. 2, 18-20 Aug. 2009, pp. 603606.

[5] Che-Wei Lee and Wen-Hsiang Tsai, "A Lossless Data Hiding Method by Histogram Shifting Based on an Adaptive Block Division Scheme", Pattern Recognition and Machine Vision, River Publishers, Aalborg, Denmark, pp. 1âĂŞ14.

[6] Soumyendu Das, Subhendu Das, Bijoy Bandyopadhyay, Sugata Sanyal, "Steganography and Steganalysis: Different Approaches", International Journal of Computers, Information Technology and Engineering (IJCITAE), Vol. 2, No 1, June, 2008, Serial Publications, pp. 1-11.

[7] Abbas Cheddad, Joan Condell, Kevin Curran, Paul Mc Kevitt, "Digital image steganography: Survey and analysis of current methods", Journal of Signal Processing, Elsevier, Volume 90, Issue 3, March 2010, pp.727-752.

[8] Sandipan Dey, Ajith Abraham, Sugata Sanyal, "An LSB Data Hiding Technique Using Prime Numbers", IEEE Third International Symposium on Information Assurance and Security, Manchester, United Kingdom, IEEE Computer Society press, USA, 29-31 Aug. 2007, pp.101-106.

[9] A. Nikolaidis, "Block-based semi-reversible data hiding without overhead information," 2013 IEEE International Conference on Multimedia and Expo Workshops (ICMEW), San Jose, CA, 2013, pp. 1-6.

[10] "SC-SIPI image database", [online]http://sipi.usc.edu/database/.