CONTENT BASED JPEG FRAGMENTATION POINT DETECTION

Qiming Li[†], *Bilgehan Sahin*[‡], *Ee-Chien Chang*[‡], *Vrizlynn L.L. Thing*[†]

[†]Cryptography and Security Department
Institute for Infocomm Research, Singapore
{*qli,vriz*}*@i2r.a-star.edu.sg*

[‡]School of Computing
National University of Singapore
{*bilgehan,changec*}*@comp.nus.edu.sg*

ABSTRACT

In the forensics analysis of raw evidence data, fragmentation point detection is crucial to differentiate fragments of evidence and identify potentially corrupted data. This need is even more prominent for JPEG images since the chance is high that an erroneous data block passes a normal JPEG decoder without triggering any errors. Therefore, it is important to verify the content of the decoded image data to determine if fragmentation and/or corruption has occurred. In this paper, we propose three different techniques for the detection of fragmentation point based on the image contents, as well as a detector built by combining these methods. We evaluate the effectiveness of these techniques and the combined detector by implementing them on a standard JPEG decoder and testing them on more than 2000 fragmented images generated from over 1200 JPEG photos.

Index Terms— Digital forensics, JPEG, fragmentation point detection

1. INTRODUCTION

When files are stored on storage devices, they are often fragmented. The description of the correspondence of the logical structure of a file and its underlying raw data fragments is typically tracked by the file system and stored separately from the file. When such information is lost, it can be very difficult to recover a fragmented file from the storage device, even when all its corresponding data fragments can still be read from it. This can happen, for example, when the file is deleted from the file system but the underlying data storage has not been recycled to store other information, or when the file system itself is damaged. Both these scenarios are of great interests to forensics applications, or to users who simply wish to recover files that were either deleted accidentally or lost due to faulty file systems. Among all files, JPEG images are of special interests due to the increasing popularity of digital cameras.

Similar to many other binary file format, the JPEG standard specifies that the first few bytes of a JPEG image must be some special sequence. This allows us to quickly find the *headers* of JPEG images among a potentially large amount of data fragments. To recover the images in their entirety in the presence of data fragmentation, we have to be able to automatically detect the point where the fragmentation occurs when scanning the data from a known header, and from there search among other data fragments instead for a better match.

The problem of fragmentation point detection is illustrated in Fig. 1. Essentially, given some partially decoded JPEG image data and a new fragment f, our objective is to determine if f is the correct fragment that follows what has already been decoded.

Fig. 1. Fragmentation detection

A naive approach to fragmentation point detection is to parse the new fragment f together with previous decoded data using a standard JPEG decoder, and see if errors and/or warnings occur during the decoding of f. However, this approach is not reliable since there is a good chance that a fragment that silently passes a standard JPEG decoder does not belong to the JPEG image in question. In fact, as shown by Pal and Memon [6], an incorrect JPEG image can be fully validated by a standard JPEG decoder. There are several reasons why this can happen. First, JPEG compressed data often looks quite random (i.e., with high entropy) and conversely, random high entropy data fragment may look like JPEG compressed data too. Second, depending on the implementation details, a JPEG decoder may choose to ignore certain errors, partly due to efficiency considerations, partly due to efforts to display

978-1-61284-350-6/11/$26.00

partial images even when a small portion of it cannot be decoded normally. Lastly, special byte sequences, for example, a fragment of all 1's or all 0's, may be deemed as part of a valid JPEG stream.

Therefore, it is important to determine if fragmentation occurs based on the actual contents of the decoded image. Intuitively, this is possible because subregions within natural images are highly consistent and any inconsistency can be easily identified by human visual system. In fact, our proposed methods are all motivated from the visual abnormalities of corrupted portions of JPEG images. However, like many other computer vision problems, it turned out to be quite challenging, especially efficiency is a major consideration due to the large number of data fragments in a typical file system.

In this paper, we propose three methods to detect fragmentation points of corrupted JPEG images. Our methods aim to detect abnormal changes in (1) DC coefficients in the DCT domain, (2) AC coefficient distributions, and (3) edges along the block boundaries. These features can be computed either locally from JPEG blocks decoded from the data fragment in question, or by combining the new data with its neighboring JPEG blocks that have been decoded previously. We then combine these methods to form a detector, which uses the results of these individual tests for final decision making.

We implement our methods by modifying libjpeg ([1]), and evaluate the methods using randomly constructed corrupted JPEG images that would cause no errors or warnings when decoded using the standard libjpeg. We show that our combined detector is able to achieve low false positives with reasonable false negatives.

We note that although our proposed methods deals with each fragment independently, more complex modeling and statistical change detection techniques, such as the sequential hypothesis testing method used by Pal et al. [7], can be used in combination with our methods to achieve better results in real world applications.

In the following, we briefly review related work in Section 2, and describe our proposed methods in Section 3. The evaluation details of our methods can be found in Section 4. We conclude in Section 6.

2. RELATED WORK

The validation based carving technique proposed by Garfinkel [5] is among the first studies that attempt to recover files from fragmented data. Garfinkel studied the statistics of fragmentation of files on real-life disk images, and showed that 16% of JPEG images are fragmented, and most of which are fragmented into three or more pieces. Once the Start-of-Image (SOI) marker and End-of-Image (EOI) marker are found for a JPEG image, the data in between is then *validated* by parsing it using a standard JPEG decompressor and see if there is an error. If there is an error, further searching is done to locate the fragmentation point. This method is computationally expensive during the searching when fragmentation occurs, since the same data may need to be decompressed several times.

Cohen [4] gave a content based JPEG fragmentation point detection using a mapping function. In essence, given a new fragment of data, an edge detection algorithm is applied on boundary between the decoded data and existing data to determine if fragmentation occurs. Their edge detection is defined as the accumulated difference between the actual pixel values along the boundary, and their estimated values computed by taking the average of the pixels from the lines above and below the boundary. The edge detector is then integrated into libjpeg to perform validation. Successful results are shown for a small number of fragmented images.

Pal et al. [7] showed that the validator approach based on standard JPEG decompressors is not sufficient by arguing that it is quite likely that an incorrect JPEG file can silently pass the validator without generating any errors. They proposed to compute a matching metric for each new fragment, which is defined as the average differences between pixels along the boundary. This metric is somewhat similar to the edge detection approach in [4]. However, they also observe that such a metric cannot distinguish correct and corrupted fragments with high certainty. Therefore, they further proposed a method for fragmentation point detection based on sequential hypothesis testing, where the decision of whether a fragmentation occurs is delayed after a few fragments have been examined.

3. PROPOSED METHODS

3.1. Main Idea

Our main idea is based on our experiences in handling corrupted JPEG images. We examine a number of corrupted images and identify image properties that change across the fragmentation point. For example, the colors or illumination levels may change (e.g., the corrupted region becomes much more green or darker). Observable changes may also include the appearances of noticeable blocking artifacts, noise-like blocks or new edges along the boundary of the corrupted image blocks. These observations are the inspirations of the proposed techniques that we describe in this section. An example of a corrupted JPEG image is shown in Fig. 2.

3.2. Changes in DC Coefficients

Our first observation is that the blocking artifacts in the first corrupted fragment becomes much more significant than what usually appears in a normal JPEG image. Since it is known that blocking artifacts are mostly caused by sharp changes in the DC coefficient, we propose to check against such changes. In fact, since the DC coefficients are difference encoded, if one DC difference in the corrupted fragment is decoded wrongly, all the DC coefficients from this point

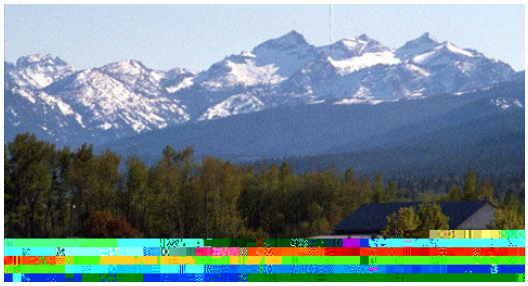

Fig. 2. Corrupted JPEG image

onwards would have incorrect values, which introduces an amplification effect that cause the visually sharp change vertically.

In particular, let $C = \{c_1, \cdots, c_n\}$ be a subset of the DC coefficients decoded from a given fragment f such that the DCT blocks that these coefficients belong to are immediately below another set of DCT blocks with DC coefficients $C' = \{d_1, \cdots, d_n\}$, which are previously decoded from other fragments. This is illustrated in Fig. 3 with $n = 7$.

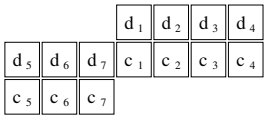

Fig. 3. Vertical DC changes

We compute the average vertical DC change of a fragment f as

$$D_f = \frac{1}{n} \sum_{i=1}^{n} |c_i - d_i|. \quad (1)$$

To effectively distinguish normal JPEG images and corrupted ones, we gather the statistics of D_f for both the correct fragments from normal JPEG images and incorrect fragments from corrupted images. After that, we set a threshold on the value of D_f based on the statistics and the required false-positive and/or false-negative rates. The actual statistics and the corresponding ROC performance can be found in Section 4.1.

3.3. Distribution of AC Coefficients

When data fragmentation occurs during the decoding of AC coefficients, we can also observe higher level of high frequency noise, compared with the statistics of natural images. An example of such noisy image blocks can be found in Fig. 4, which is a close-up look of the corrupted region of the image shown in Fig. 2.

Noisy images blocks like these correspond to AC coefficients that have higher energy in the mid-to-high frequency region, which we define as frequency band from the 33-rd to the 64-th DCT coefficients in the zig-zag order.

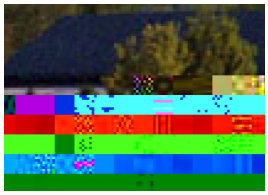

Fig. 4. Noisy image blocks

To differentiate normal AC coefficients and noisy ones, we find the distributions of mid-to-high frequency AC coefficients from normal JPEG images. Given a new fragment f that is decoded into n DCT blocks, for each DCT block b_i $(1 \leq i \leq n)$ we examine the absolute values of the AC coefficients.

We say that an AC coefficient at the j-th location in the zig-zag order is *large* if its absolute value is larger than 97% of the AC coefficients at the same location in normal JPEG fragments.

We count the number A_i of AC coefficients in the mid-to-high frequency band that are large, and we take the maximum value A_f of A_i over n DCT blocks as the AC statistics of the fragment f. In other words, we have

$$A_f = \max_{1 \leq i \leq n} (A_i). \quad (2)$$

We further gather statistics of the value A_f over normal JPEG fragments and that in corrupted fragments to determine the performance of the AC test in terms of ROC (Section 4.2).

3.4. Edge Density

In many of the corrupted images, horizontal edges are clearly seen at the boundary of the corrupted DCT blocks. An example of a corrupted image and its edges are shown in Fig. 5. As expected, horizontal edges are concentrated in the corrupted region.

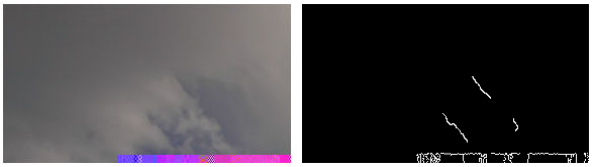

Fig. 5. Edges in corrupted JPEG image

Therefore, if we run an edge detection algorithm on the boundary pixels when the fragment is not the right one, we should be able to detect an edge with high probability. In fact, the mapping function proposed by Cohen [4] is essentially an edge detector. Some other methods [7, 6] are similar, and can be considered as simple edge detectors as well.

We use the OpenCV [2] implementation of the Canny edge detector [3] for our purpose, which is widely used for

many computer vision applications. This edge detector assigns a value to each pixel of the input image to indicate if there is an edge and the direction of it.

For a given fragment f, assume that there are n pixels along the boundary of the DCT blocks that are next to previously decoded DCT blocks from other fragments. We define $p_i = 1$ $(1 \leq i \leq n)$ if the edge detector detects a horizontal edge at the i-th pixel along the boundary, and $p_i = 0$ otherwise. We further define the *edge density* E_f of fragment f as

$$E_f = \frac{1}{n} \sum_{i=1}^{n} p_i. \tag{3}$$

Similar to the other two tests, we gather statistics of E_f from both normal fragments and corrupted fragments, and examine its performance in terms of ROC (Section 4.3).

3.5. Combined Detection

To form a final decision as whether a given fragment is the correct fragment that follows a given (partially decoded) JPEG image, we combine the three proposed detection methods together.

In particular, we assign a weight for each of the detection values of the three proposed methods, and compute a weighted sum as the final detection value, which is then compared with a threshold for making the final decision. That is, the final "score" for a given fragment f is given by

$$S_f = \alpha D_f + \beta A_f + \gamma E_f \tag{4}$$

where $\alpha + \beta + \gamma = 1$, and D_f, A_f and E_f are defined in equations (1), (2) and (3) respectively.

4. EVALUATIONS

To evaluate our proposed methods, we implement them on top of the libjpeg library, and randomly generate corrupted JPEG files using a database that contains over 1200 public domain photos of natural scenes (`http://www.pdphoto.org`). We use these photos to generate statistics of "normal" JPEG images.

Next, for each JPEG file, we divide it (logically) into blocks of 512 bytes, and randomly select a fragmentation point that is beyond its first SOS marker, which ensures that fragmentation occurs in the compressed data stream and not in the "header" part. Next, we randomly select another data block from all other files, append the selected data block to the JPEG file at the selected fragmentation point to create a corrupted JPEG image.

The resulting image is then parsed by the standard libjpeg decoder, and it is only accepted when no errors or warnings is generated from libjpeg, except the one warning that indicates premature ending of the JPEG image. In other words, we only deal with corruptions that cannot be detected by a standard decoder. We generated over 2100 corrupted photos as our test images. The statistics of "corrupted" fragments are obtained from the corrupted regions of these generated images.

For each generated corrupted JPEG image, our detectors process it block by block to find a fragmentation point. If no fragmentation is reported when the actual fragmentation point is reached, it is counted as a false-negative, and if the reported fragmentation point is before the actual one, it is counted as a false-positive. We then vary the threshold to find the ROC curves.

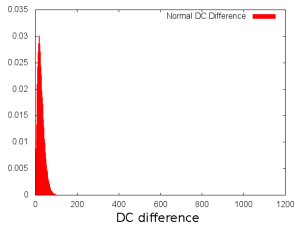

Fig. 6. DC changes in normal JPEG images

4.1. Statistics of Vertical DC Changes

The statistics of DC changes in the vertical direction in normal JPEG images is obtained by examining the DC changes in all the fragments in the original images in our database. The result is as shown in Fig. 6.

The statistics of DC changes in corrupted images is computed from the corrupted fragments of all the generated images. The result is as shown in Fig. 7.

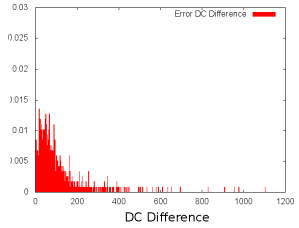

Fig. 7. DC changes in corrupted JPEG fragments

We build a detector using only the DC statistics and its performance is as shown in Fig. 8.

4.2. Distributions of AC

For AC coefficients, we first gather histograms for all the mid-to-high frequency components, and for an AC coefficient a_i at the i-th position in the DCT block, we set a threshold δ_i, such that the probability $\Pr[a_i > \delta_i] < 0.03$, where 0.03 is another empirical threshold that we used throughout our experiments.

After that, if an AC coefficient a_i in a given DCT block is greater than δ_i, we say that it is *suspicious*. Then we find

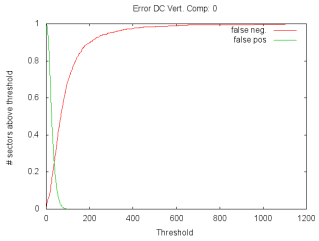

Fig. 8. ROC curve for detector based on DC changes

the statistics of suspicious AC coefficients in a random block from a normal JPEG fragment through experiments, which is shown in Fig. 9.

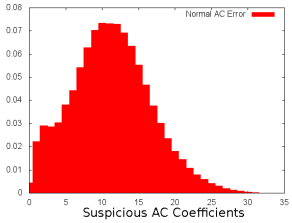

Fig. 9. Suspicious AC coefficients in normal JPEG images

Similarly, we find the statistics of suspicious AC coefficients of corrupted fragments from the test images we generated, which is shown in Fig. 10.

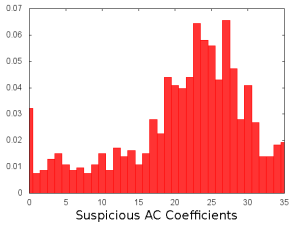

Fig. 10. Suspicious AC coefficients in JPEG fragments

We build a detector using only the AC statistics and its performance is as shown in Fig. 11.

4.3. Statistics of Edge Density

We build the histograms for edge density as defined in Section 3.4 for normal JPEG fragments and corrupted JPEG fragments, which are as shown in Fig. 12 and Fig. 13 respectively.

We build a detector using only the edge density and its performance is as shown in Fig. 14.

4.4. Combined Detector

To evaluate the combined detector (Section 3.5), we try many possible combinations of the weights to search for the optimal

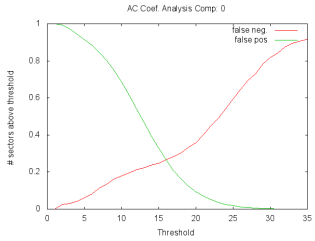

Fig. 11. ROC curve for detector based on AC statistics

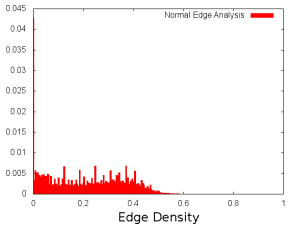

Fig. 12. Edge density in normal JPEG images

values for the three different tests. The optimal weights we found are as follows.

$$\alpha = 0.40, \quad \beta = 0.45, \quad \gamma = 0.15. \tag{5}$$

The performance of the combined detector when the weights in (5) are used is shown in Fig. 15.

As we can see, the combined detector has an equal-error-rate (ERR) that is slightly larger than 0.2. From this figure, we can also determine the trade-off between the false-positive rate and the false-negative rate. For example, if we require a small false-positive rate of 5%, we can see from Fig. 15 that the corresponding false-negative is slightly less than 0.4, where the threshold value is about 0.3.

5. FUTURE WORK

In this paper we show that it is possible to make use of various statistics of the decoded image content to distinguish between a normal JPEG fragment from a corrupted one. We are working on improving the detection results from these statistics, while at the same time devising better statistics as the measure of corruption.

We note that two of our three proposed methods rely only on decoding results in the DCT domain, which means that these methods do not require full decompression of the JPEG image. This approach is favorable when processing speed is a major consideration, since the inverse DCT operations in JPEG decompression is, in general, the most computationally expensive step. Our future plan includes further exploration of reduction of computational costs of fragmentation detection.

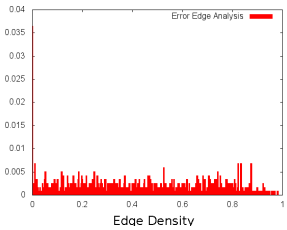

Fig. 13. Edge density in JPEG fragments

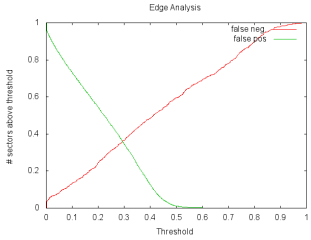

Fig. 14. ROC curve for detector based on edge density

We note that our techniques can be, and should be, used together with other statistical techniques to obtain better performance. In general, the fragmentation point detection problem is a special case of statistical change detection, and it is possible to utilize existing statistical tools. In fact, it is shown by Pal et al. [7] that sequential hypothesis testing can be used effectively in combination with fragment statistics. We will investigate how such techniques can be integrated into our methods.

Lastly, in some of the previous work, the evaluation of fragmentation techniques is typically done in a more or less ad-hoc manner by using a small set of test images. In this paper, we have employed a moderately large number of randomly fragmented images for testing. In the future we are planning to design more systematic methods to generate test sets that would resemble fragmentation and corruption that would occur in real world file systems.

6. CONCLUSIONS

In this paper, we study the problem of fragmentation point detection for JPEG images. In contrast with some previous techniques, which rely heavily on finding invalid JPEG code streams, we acknowledge that it is very likely that a high entropy corrupted fragment can be part of a valid JPEG stream, and focus on using the statistics of the image content to determine the fragmentation points.

We propose three different techniques, which are all motivated through observations of actual corrupted images, and evaluate their performances through experiments with real-life JPEG photos and corrupted images that are randomly generated from them. We then find the optimal weights to combine the three methods together to form a integrated detector, and analyze its performance through experiments.

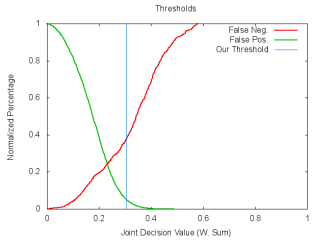

Fig. 15. ROC curve for the combined detector

We also note that there are various ways to improve our current results, including the integration with change detection techniques such as hypothesis testing.

7. REFERENCES

[1] Independent JPEG group's free JPEG software. `http://www.ijg.org/`.

[2] Gary R. Bradski and Vadim Pisarevsky. Intel's computer vision library: Applications in calibration, stereo, segmentation, tracking, gesture, face and object recognition. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, volume 2, page 2796, 2000.

[3] John F. Canny. A computational approach to edge detection. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 8(6):679–714, 1986.

[4] Michael I. Cohen. Advaned Jpeg carving. In *Proceedings of the 1st international conference on Forensic applications and techniques in telecommunications, information, and multimedia and workshop (e-Forensics)*, January 2008.

[5] Simson Garfinkel. Carving contiguous and fragmented files with fast object validation. In *Digital Forensics Research Workshop*, volume 4S of *Digital Investigation*, pages S2–S12, 2007.

[6] Anandabrata Pal and Nasir Memon. The evolution of file carving. *IEEE Signal Processing Magazine*, 26(2):59–71, March 2009.

[7] Anandabrata Pal, Husrev T. Sencar, and Nasir Memon. Detecting file fragmentation point using sequential hypothesis testing. In *Digital Forensics Research Workshop*, volume 5 of *Digital Investigation*, pages S2–S13, 2008.