Identification of JPEG files fragments on digital media using binary patterns based on Huffman code table

Alexander Sorokin
HSE MIEM
National Research University Higher School of Economics
Moscow, Russia
asorokin@hse.ru

Ekaterina Makushenko
HSE MIEM
National Research University Higher School of Economics
Moscow, Russia

Abstract— **File fragmentation proves to be a major challenge for the majority of file carving techniques. Following the works of Simson Garfinkel, Nasir Memon and other authors we seek to find a technique to identify digital fragments (clusters or sectors) of JPEG-files on the digital storage medium or at least sort all the fragments of the storage based on their probability of being the part of JPEG-file from the most probable to the least probable. This paper offers the technique of identifying clusters of JPEG-files on the storage medium based on binary patterns and the experimental results of an attempt to build a similar technique for sector identifying.**

Keywords—file recovery; fragmentation; JPEG; Huffman code table

I. INTRODUCTION

Basic carving techniques as described in [1] are strongly based on the assumption that the files are allocated contiguously. They rely on special sequences of bytes identifying the beginning of the file and in some cases the end of the file as well. The very basic file carving method is just to merge all storage space between such byte sequences and consider it as a recovered file. The main problem with that is the phenomenon called fragmentation which is also described in [1]. If a file that is to be recovered is fragmented the aforementioned technique will return an incorrect result as the storage space between actual file fragments will be included into the recovered file as well. Such a challenge led to the researches focused on finding a solution that would allow carving of fragmented files with sufficient completeness and accuracy. Several researches such as described in [2] – [4] were made and some, specifically [4] were considering the recovery of fragmented JPEG files. Two reasons may be named for such a focusing. The first one is that JPEG file format is among the most popular file formats and the second one is that it is also an encoded file format with quite a complex encoding algorithm thus challenging for separate parts processing. That is why special methods considering only that file format are still demanded and those described in this paper may be placed among them.

One of such methods is to find the point where fragmentation starts as describes in [3]. After that the search for the next file fragment is done. We consider it to be useful for that and other possible techniques that require complete testing of data storage medium memory to first identify the fragments of the memory that belong to the file type that matches the type of a file being recovered. Because of aforementioned reasons we made the research for JPEG file format.

II. IDENTIFIYNG CLUSTERS OF JPEG FILES

Following the ideas of [4] we have decided to build binary patterns based on Huffman code tables. In our case such patterns are to be used for identification of clusters belonging to JPEG files with standard Huffman code tables specified in [5].

A. Cluster-level fragmentation

As described in [1] and [4] files are allocated by the file systems by the minimal chunks of data called clusters. Hereinafter we will consider the size of cluster as 4KB.

During the allocation process a file system can face one the following circumstances that would result in file fragmentation.

First is the lack of free space on the storage medium for contiguous allocation of file. If there are only separate groups of free clusters the next file allocated will obviously be fragmented and the allocation algorithm may vary between file systems. For example the allocation can start from the largest

consecutive group of clusters or from the cluster with the smallest number starting from the beginning of the medium. The first scenario would result in the least possible number of fragments but may also allocate the last fragment of the file into the clusters with the smaller numbers than its first fragment. The second scenario is free from such a risk but may result in a large number of small fragments. The main popular file systems however seems to use combinations of these two scenarios avoiding allocating file so that its earlier parts have the larger cluster numbers then its later parts. Still Pal and Memon states in [1] that most file systems would allocate the beginning of the file in the largest consecutive group of clusters.

Second is the edition of previously saved files which is quite obvious. When a file is saved before another file is allocated and then is once again opened for writing it may require additional storage space as its size has increased. At the same time the next consecutive cluster after its last cluster may be already occupied by another allocated file. In this case additional file data can be allocated somewhere else on the medium resulting in file fragmentation. Pal and Memon in [1] give examples of special techniques used by different file systems for reducing fragmentation under such circumstances but conclude that such techniques can not eliminate fragmentation completely.

Additionally they mention that some file systems may force fragmentation due to some optimizations or other reasons.

It can be also noticed that Garfinkel in [2] has shown that the fragmentation of user files which are the main aim of recovery procedure in most cases is high. The general idea of fragmentation and its affection on file carving is shown on Fig.1.

B. *Selecting the Huffman codes for patterns*

In order to build binary patterns for JPEG clusters identification we have formed an array of 100 JPEG files made with to 10 cameras and studied the Huffman codes distribution on this array of files. Of four Huffman code tables specified in JPEG file format [5] we have chosen the luminance AC coefficients table. Our criteria for the choice were the following. First we planned to have at least one long code in our pattern in order to lower the false positive error rate. Amongst Huffman code tables only aforementioned table had the minimal percentage of longest codes in a single file more than zero, as shown in Table 1. That means that for every other table there were files in our array where the longest codes were not present at all. Second, that minimal amount for luminance AC coefficients table was still not less than the amount of clusters in our biggest file that made it possible that each and every cluster really contains at least one of such codes.

C. *Building the binary patterns*

Denote by s any binary string that can be viewed as a pattern for possible search in a cluster under examination. Denote four Huffman code tables defined in [5] as $H_{i,j}$ where i is the index corresponding with the component type with "0" for Y luminance components and "1" for Cb or Cr components and j is the index corresponding with the coefficients of said coefficients – "0" for DC coefficients and "1" for AC coefficients. For example the luminance AC coefficients table will be denoted as $H_{0,1}$. Denote a Huffman code from Huffman code table $H_{i,j}$ as $h \in H_{i,j}$, denote by $d(h)$ uncompressed bits following Huffman code h and concatenation operation by "▪" symbol. If the length of the code h is important denote by h^k the Huffman code of length k For our method we have used patterns described by the following formula given in our earlier paper [6]:

$$s_t = h_i^2 \bullet d(h_i^2) \bullet h_j^{16} \bullet d(h_j^{16}) \bullet h_k^2 \quad (1),$$

where s_t is the t-th pattern, h_i, h_j and h_k – independently chosen Huffman codes (hence different subscript indexes) of all Huffman codes of respective length, $h_i, h_j, h_k \in H_{0,1}$. We concatenate one of 2 Huffman codes of length 2 from luminance AC coefficients code table taken together with its uncompressed bits with one of 125 Huffman codes of length 16 from the same code table taken together with its uncompressed bits as well. Then we concatenate the resulting string with one more separately chosen Huffman code of length 2 from the same code table. The total amount of patterns is 500 that is 2 variants for codes of length 2 squared as we use such codes twice multiplied by 125 which is the amount of codes of length 16.

D. *Setting the criteria*

In order to identify clusters belonging to JPEG files we offer to search the whole set of patterns in every cluster of data storage medium. And here we need to specify the criteria for accepting the cluster as the one we need or reject it. First, as we made our patterns of Huffman codes from JPEG files, we expect that at least some of these patterns to be present so the first criterion would be

$$\sum_{i=1}^{500} p_i > 0 \quad (2),$$

where p_i is the number of occurrences for the pattern number i (i runs from 1 to 500 through the whole set of patterns).

This criterion being used alone is too weak in rejecting the clusters which are not parts of JPEG files, so we have added the upper border as well:

$$\sum_{i=1}^{500} p_i < 20 \quad (3),$$

or combining these two borders together

$$0 < \sum_{i=1}^{500} p_i < 20 \quad (4).$$

This means that we search all the patterns in every cluster and calculate the sum of occurrences of all patterns. In case that sum is both positive and less than 20 we accept the cluster as the part of JPEG file with default Huffman code tables.

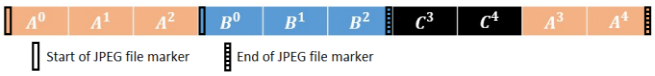

Fig.1. Fragmentation. File A is allocated into two fragments and needs special carving techniques dealing with fragmentation. File B is contiguous and can be carved by simply finding start and end of file markers.

TABLE 1. Huffman codes distribution in JPEG files

Huffman code table	Values	Huffman codes length					
		7	8	9	14	15	16
Luminance DC coefficients	Min.	0,00%	0,00%	0,00%			
	Avg.	3,42%	0,95%	0,07%			
	Max.	13,30%	3,89%	0,54%			
Luminance AC coefficients	Min.	2,29%	0,93%	0,68%		0,00%	0,03%
	Avg.	4,47%	2,02%	1,15%		0,00%	0,19%
	Max.	6,72%	3,37%	1,77%		0,02%	0,44%
Chrominance DC coefficients	Min.	0,00%	0,00%	0,00%			
	Avg.	1,48%	0,45%	0,09%			
	Max.	10,31%	4,35%	0,86%			
Chrominance AC coefficients	Min.	0,02%	0,04%	0,00%	0,00%	0,00%	0,00%
	Avg.	0,64%	1,57%	0,54%	0,03%	0,02%	0,01%
	Max.	1,98%	2,97%	2,41%	0,32%	0,28%	0,14%

E. Error rates

Using the aforementioned patterns the way we have described above gives us the following quality of identification with the null hypothesis of "the cluster is a part of a JPEG file". Type II error (i.e. the probability of a cluster being mistakenly rejected) is about 0.03. This rate was experimentally achieved twice on two separate sets of JPEG files, as described in earlier papers [6] and [7].

In [7] type I error (i.e. the probability of a cluster being mistakenly accepted) was also experimentally tested for different file types. We had several file types for that we had said rate very low (less than 0.05) – these were RTF, XML, CPP, for several others we had that rate less than 0.3 – these were TXT, HTML, MPEG. We only name here user files which are usually of most interest during recovery.

F. Possible applications

We can see two main possible applications for cluster identification.

The firs one is pre-recovery identification of clusters in order to accelerate the work of any algorithm requiring complete testing. For that purpose all the clusters on digital storage medium is tested with the aforementioned technique and are divided into two parts according to the criteria – those that were accepted and those that were rejected. Starting each testing with the cluster from the "accepted" group would allow the search to be completed within that group in most cases. The rate of acceleration would depend on what file types were present at the media along with the file under recovery. In case of JPEG file recovery one can expect that almost all the clusters of RTF, CPP, XML files and the majority of clusters of MPEG, HTML, TXT and some other files would be excluded from the search.

The second possible application is the post-carving processing of incorrect carving results. In case a file was allocated in several fragments keeping the proper order of its clusters and after that it was carved from its beginning to its several fragments of other files. We have made a software prototype that would allow the user to exclude separate clusters or the sequences of clusters and then see how the file would look like if processed without them. Part of its file processing window with user interface is shown on Fig. 2. The program can apply our identification technique and return the list of clusters that were rejected. The user can manually try excluding them or add more clusters in order to find the best representation of an image being recovered.

III. IDENTIFIYNG SECTORS OF JPEG FILES EXPERIMENTS

Cluster-level approach means the assumption that each and every cluster is physically contiguous and the fragmentation can only appear during file allocation by file system. Meanwhile that is not guaranteed in every case.

First, Pal and Memon in [1] mentioned wear-leveling algorithm as the possible reason for the fragmentation to occur on the lower lever making some clusters not physically contiguous. Wear-leveling means that the firmware which is being supplied with the digital storage medium enforces the equal use of all physical memory units on the medium. As some of them are used more than others this algorithm may remap logical addresses making logically sequential memory units physically lacking such a quality. If a disk image of such a medium with damaged controller is made then its memory units would be read from the smallest to the largest estimated logical addresses without actual possibility to check their logical addresses.

Second, the very similar situation may occur in case one or several physical memory units are considered by the storage medium controller as bad sectors. The controller may act similar to wear-leveling algorithm making it hard or impossible to verify physical data sector's logical address in case the controller is damaged.

Taking these possibilities into account we have made several attempts to scale our technique for cluster-based identification for sector-level. Hereinafter we will consider the size of sector as 512 Bytes.

A. Using the set of patterns built for cluster identification

Our first attempt was to use our pattern set for cluster identification in sector identification without any modification built according to (1). Since a cluster is 4 KB and a sector is only 512 Bytes a sector is 8 times smaller than a cluster is. That means that in order our criteria to work properly (and most precisely its lower border) every cluster successfully identified must have at least 8 occurrences of our patterns. This is so as each cluster is now processed as 8 separate parts and each and every such part needs to include at least 1 occurrence to be accepted. We have made an estimation of II type error rate by finding the following value. In case a cluster contains 8 occurrences or more it is added to the sum as 8 (as all 8 of its sector can potentially be accepted) and if it contains

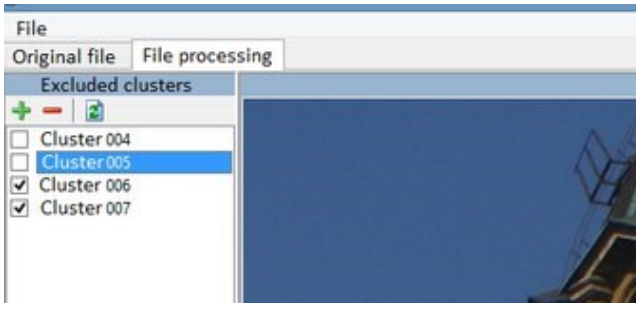

Fig.2. Part of file processing window in post-carving processing software. Clusters list on the left is made by program and can be edited by the user. Separate clusters can be selected and the image on the right shows the view of the original file processed without selected clusters.

less occurrences it is added to the sum with its number of occurrences (as no more sectors of that value can be accepted). The final sum was then divided by the number of clusters multiplied by 8. Our result was about 0.71 that means that type II error rate is expected to be no less than 0.29 which is far worse than on the cluster level but still can be used for some applications given that type I error rate remains low. Our experimental result for such case was only 0.51 which is even less and thus had to be increased. Considering type 1 error rate the following can be said. First the rate of sectors rejected by the lower-border criterion was no worse than for cluster level identification since we searched the very same set of patterns in only the fractures of binary strings that were examined on cluster level. Second the rate of sectors rejected by the upper-border criterion could be lowered for the same reasons as with the first criterion. Thus we had to make the second criterion stronger so we consecutively moved the upper border from 20 to 19, 18 and so on up to 5 where the rate of sectors successfully accepted became less than 0.5. The last value was clearly unsatisfactory but since the rate decreased really slowly we considered it possible to actually lower the upper border significantly.

B. Using the modified set of patterns and modified criteria

In order to increase the rate of successfully accepted clusters we had to modify our patterns somehow. One of possible modifications was "pruning" them by removing the last Huffman code of length 2 and the uncompressed bits after the code of length 16 so that new patterns looked like this:

$$s_t = h_i^2 \bullet d(h_i^2) \bullet h_j^{16} \quad (5)$$

Because of lack of one code of length 2 for what there were 2 possible values in AC luminance coefficients code table there were only 250 codes instead of 500.

With the patterns built according to (5) and the criteria (4) we have achieved the rate of correct identification of sectors belonging to JPEG files of about 0.73 that makes the type II error rate of about 0.27 which is even slightly better than our theoretically estimated level of 0.29 that was claimed acceptable.

In order to make our criteria even stronger in rejecting the sectors of other file types we have decreased its upper border to the level of 10. This made our criteria look like this:

$$0 < \sum_{i=1}^{250} p_i < 10 \quad (6).$$

Processing storage media sectors under (6) gave us the type II error rate of about 0.28 which is still acceptable. The type I error rate depends of file types sectors of which are present and finding its precise value in different cases is the goal of our following research.

IV. POSSIBLE REASONS FOR FILE PARTS IDENTIFICATION DEMAND

The most obvious possible application for the aforementioned results is the enhancement of different file carving algorithms that deal with fragmented files and require complete testing of possible file data. Taking into account Garfinkel's results in [2] and the storage media size growth the fragmentation because of lack of storage space can be expected to become a rarer event. Taking this into account the actual demand for identification techniques may be questioned.

With the growing amount of file carving software and publications describing the algorithm of data recovery we find it possible that a special kind of software can appear. If the user wants his personal data to be deleted with the minimal chance of its recovery then user's files can be allocated so that the majority of file carving techniques or at least the most basic of them would fail to recover any private data. This can be achieved with no need to actually rewrite the memory units with ones and zeroes which may require significant time. A user seeking to keep his personal files confidential may use the combination of a quick way to corrupt file system data (which is expected to take much less time than doing similar procedure with the whole storage medium) and the special way of allocating files. We can suppose that the logical addresses of a storage medium can be purposely remapped much like the sectors under the influence of the way wear-leveling algorithms or bad sectors remapping algorithm are. The fact that wear-leveling algorithm do not create any significant troubles for users be that unsatisfactory low access speed or any file operating errors makes it possible to expect that such an anti-carving algorithm would not create them either.

In case such an algorithm exists and allocates files choosing clusters for allocation randomly from the whole set of free clusters then recovery of every file will require using of special techniques dealing with highly fragmented files. Pre-carving identification that can reduce the amount of clusters to be tested can significantly increase their speed of operation.

V. CONCLUSION

In this paper we have offered techniques for identification of JPEG files clusters and sectors. These techniques can be used in two ways.

First all the data blocks of data storage medium can be processed separately and divided into two parts – likely belonging to JPEG files and unlikely belonging to JPEG files. That pre-carving procedure is highly parallelizable and can be implemented on multiprocessor systems such as GPU which will be done during our following research.

Second it can be used for creating the post-carving software that would be able to reject clusters of other files from incorrectly carved JPEG files.

REFERENCES

[1] A. Pal, N. Memon, The Evolution of file carving, IEEE Signal Processing Magazine, vol. 26, issue 2, 2009, pp.59–71.

[2] S. L. Garfinkel, Carving contigious and fragmented files with fast object validation, Digital Investigation, vol. 4S, 2007, pp. S2–S12.

[3] A, Pal, H. T. Sencar, N. Memon, Detecting file fragmentation point using sequential hypothesis testing, Digital Investigation, vol. 5, 2008, pp. S2–S13.

[4] H. T. Sencar, N. Memon, Identification and recovery of JPEG files with missing fragments, Digital Investigation, vol. 6, 2009, pp. S88–S98.

[5] CCITT T.81 (09 /92) Digital Compression and Coding of Continuous-tone Still images. The international telegraph and telephone consultative committee, ITU.- 1993.

[6] A. Sorokin, On differentiation of clusters belonging to JPEG files, *Problemi informatsionnoy bezopasnosti. Kompjuternie sistemy* [Problems of information security. Computer systems], 2012, vol. 4, pp. 61–67 (in Russian).

[7] N. Kostylova, A. Sorokin, On elaboration of the method of identifying clusters belonging to JPEG files, *Bezopanost' informatsionnih tehnologiy* [Information technologies ecurity], 2014, vol. 4, pp. 53–58 (in Russian).