THE JOURNAL OF DIGITAL FORENSICS, SECURITY AND LAW

Journal of Digital Forensics, Security and Law

Volume 13 | Number 2 Article 7

October 2018

Fingerprinting JPEGs With Optimised Huffman Tables

Sean McKeown
Edinburgh Napier University, s.mckeown@napier.ac.uk

Gordon Russell
Edinburgh Napier University, g.russell@napier.ac.uk

Petra Leimich
Edinburgh Napier University, p.leimich@napier.ac.uk

Recommended Citation
McKeown, Sean; Russell, Gordon; and Leimich, Petra (2018) "Fingerprinting JPEGs With Optimised Huffman Tables," *Journal of Digital Forensics, Security and Law*: Vol. 13 : No. 2 , Article 7.
DOI: https://doi.org/10.15394/jdfsl.2018.1451
Available at: https://commons.erau.edu/jdfsl/vol13/iss2/7

Fingerprinting JPEGs With Optimised Huffman Tables

Cover Page Footnote
This research was supported by a scholarship provided by Peter KK Lee.

FINGERPRINTING JPEGS WITH OPTIMISED HUFFMAN TABLES

Sean McKeown, Gordon Russell, Petra Leimich
School of Computing
Edinburgh Napier University, Scotland
{S.McKeown, G.Russell, P.Leimich}@napier.ac.uk

ABSTRACT

A common task in digital forensics investigations is to identify known contraband images. This is typically achieved by calculating a cryptographic digest, using hashing algorithms such as SHA256, for each image on a given medium, and comparing individual digests with a database of known contraband. However, the large capacities of modern storage media and time pressures placed on forensics examiners necessitates the development of more efficient processing methods. This work describes a technique for fingerprinting JPEGs with optimised Huffman tables which requires only the image header to be present on the media. Such fingerprints are shown to be robust across large datasets, with demonstrably faster processing times.

Keywords: digital forensics, image comparison, image processing, known file analysis, partial file analysis

1. INTRODUCTION

Digital Forensics is developing at a rapid pace to keep in step with advances in technology. Approaches which worked well decades ago may not remain effective, as the nature of the underlying evidence shifts, or as datasets surpass the terabyte scale (Beebe & Clark, 2005). The nature of the field demands ongoing research, with gaps existing in prominent in areas such as triage and data reduction (Quick & Choo, 2014).

One common aspect of many public sector investigations is the detection of contraband material within a large number of images. Traditionally, this is achieved using file fingerprints generated by means of cryptographic hashing (Garfinkel, Nelson, White, & Roussev, 2010). The fingerprints extracted from a piece of evidence are compared with a database containing fingerprints of known files of interest. These fingerprints require that the entire file be processed, incurring IO and CPU overhead, and are sensitive to any change in the binary data.

However, if an equivalent fingerprint could be created from information residing near the beginning of a file, then significant performance improvements could be gained by avoiding the IO required to process the whole file with more traditional fingerprinting techniques. Even if this partial file technique created fingerprints with a larger collision domain, false positive matches could be additionally verified by a full file hash, and so performance gains with such a combined hashing approach is still effective as long as the false positive rate was reasonably low.

The main contribution of this paper is an inexpensive method for creating fingerprints of JPEG files. Huffman tables are present in the header of every JPEG, with the fingerprinting method in this work exploiting Huffman tables which have been optimised for maximal compression, which is an increasingly common encoding technique used on the Web. An analysis of the distinctness of such fingerprints is provided, as well as an examination of the portion of the file which must be processed to extract them. Finally, the method

is evaluated with timed benchmarks comparing the extraction process to a traditional hashing method.

This paper demonstrates that the proposed fingerprinting technique is faster than traditional hashing techniques on a large dataset of real world images. In analysing the false positive rate, the collision rate was shown to be almost zero.

When Huffman tables are optimised for the content of the image, the table communicates coarse information about the image, while changes to EXIF metadata and some small image manipulations have no affect on the table. This work is timely, as there is a technological trend towards optimally encoding images in the JPEG format, such as those published recently by Mozilla (Mozilla, 2017) and Google (Alakuijala et al., 2017).

Experiments are carried out using three datasets: Flickr 1 Million (Huiskes, Thomee, & Lew, 2010), containing 1 million JPEGS; Govdocs (Garfinkel, Farrell, Roussev, & Dinolt, 2009), containing 100,000 JPEGs; and a preprocessed version of Govdocs with optimised Huffman tables produced by the authors.

2. JPEG COMPRESSION OVERVIEW

The JPEG standard (Wallace, 1992), is a lossy compression technique for reducing the file size of images. The standard leverages properties of human vision in order to provide the best trade-off in perceived image quality to compression ratio, with several stages of compression being utilised.

During compression, images are typically converted to the `YCbCr` colour space, separating the luminance and chrominance channels, the latter of which may be optionally sub-sampled. The result is then divided in to 8×8 pixel blocks, which are transformed to the frequency domain using the Discrete Cosine Transform (DCT) to produce a matrix of 64 coefficients. The coefficient at the top left of the matrix, known as the DC (Direct Current), represents the mean colour value of the block, while the remaining 63 coefficients, which are known as AC (Alternating Current), contain horizontal and vertical frequency information. As most of the human sensitive aspects of the signal are concentrated in the top-left of the matrix, which hosts the low frequency coefficients, much of the higher frequency information may be discarded, or represented more coarsely. This is achieved by quantization, with the JPEG quantization table mapping the relative compression ratios of each DCT coefficient. The standard provides a generic quantization table, which can be scaled to vary image quality.

The quantization process results in many AC coefficients becoming zero. A run-length encoding scheme is then applied which compresses these runs of zeroes efficiently. Additionally, as the average colour (DC) of each 8×8 block is expected to change gradually throughout the image, differential coding is used to efficiently compress the colour differences between blocks. The coefficient compression utilises variable length encoding schemes, with the data stored as a set of bit length and value pairs. This information is further encoded using single byte codes, which in turn represent the magnitude of the DC, or combined magnitude and run-length for the AC coefficients. As these codes are repeated frequently, Huffman encoding can be used to compress their representation to variable length bit strings. This allows for frequently occurring codes to be represented in perhaps two or three bits, instead of a byte. The JPEG standard provides a default mapping of these Huffman bit strings for the AC and DC byte codes. However, for more efficient compression, a per image optimised Huffman table may be generated based on the actual occurrences of these codes, resulting in smaller file sizes.

Figure 1 depicts the beginning of a sample JPEG image. Immediately following the JPEG start marker are the application markers which specify the particular JPEG form format (such as JFIF, EXIF) and miscellaneous metadata, such as title, comments, camera settings, camera model, or editing software information. This is then followed by decompression information comprised of the quantization and Huffman tables. The most basic (baseline) JPEG makes use of two quantization tables, one for luminance,

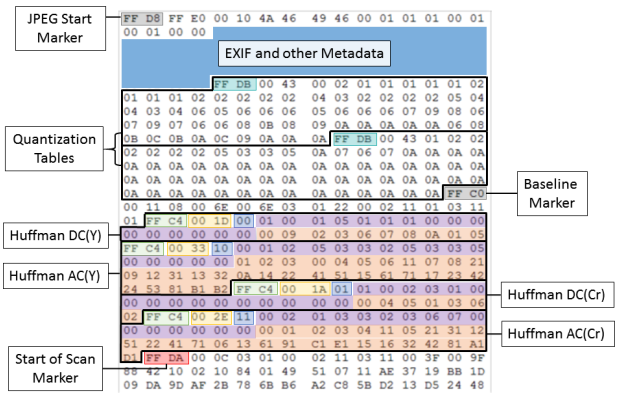


Figure 1: **The structure of a sample JPEG as it is stored on disk. The metadata section may be long, and it is abbreviated here for clarity.**

and another for chrominance, with four Huffman tables for the combinations of AC/DC and luminance/chrominance. This is then followed by the actual image scan data, which is stored sequentially, from the top of the image to the bottom. An alternative format, the progressive JPEG, may contain multiple image scans, starting with low resolution versions of the image and increasing in steps, allowing for images to increase in quality as they are loaded on the Web. Progressive JPEGs may contain more Huffman tables, which are used for each individual scan.

Mozilla's MozJPEG (Mozilla, 2017) introduces tweaks to the encoding process by modifying the original JPEG libraries while remaining compliant with the specification. The technique uses optimised Huffman tables. All images are converted to the progressive JPEG format, and new quantization table presets are provided to better accommodate high resolution images. Google's Guetzli (Alakuijala et al., 2017) takes a more aggressive approach, with coarse quantization presets, Huffman table optimisation and post processing the DCT coefficient matrix. Guetzli produces sequential images, rather than using progressive JPEGs.

3. RELATED WORK

This section outlines existing work in cryptographic file hashing in order to detect known files, before exploring work focusing on the analysis of JPEG header features.

3.1 Forensic File Hashing

Cryptographic hashes are fundamental to the digital forensics process, deployed as a mechanism for verifying the integrity of the data, subject to chain of custody assurance, as well as to fingerprint both known good and known bad files for later database lookups. J. Kornblum (2006) noted that such hashes, which are typically based on the entire content of a file or media, can easily be attacked by modifying a single bit in the original data, which produces an entirely different hash digest. Such changes may be used to interfere with automatic detection processes. In order to mitigate this, piecewise hashes may be used, where data is split in to chunks which are hashed separately, such that the fingerprint it is more resistant to manipulation. J. Kornblum (2006) extends prior piecewise hashing methods by applying a rolling hash system. This is incorporated in to the ssdeep tool, which provides a conservative estimate of the number of identical bytes in a file, providing a similarity score from 0–100.

Piecewise hashing can give false positives due to the existence of common data blocks in various file types. Roussev (2010) describes sdhash, a method for selecting statistically improbable features when producing data fingerprints, while Garfinkel et al. (2010) focus on reducing the number of common non-distinct blocks from the hash database.

Breitinger et al. (2013) compare and contrast the properties of full file cryptographic hashing, bytewise approximate matching, and semantic approximate matching. Binary methods were shown to be much faster, though much less resistant to content preserving modifications, while out performing semantic methods in the realms of damaged or embedded file detection. The authors suggest that the relative merits of each method should be exploited with an ordered approach. Traditional file hashing can be used to flag up obvious contraband, followed by costly semantic methods to detect any modified images. Finally, damaged or embedded fragments may be

detecting using bytewise approximate matching.

McKeown, Russell, and Leimich (2017) exploit image encoding metadata and small blocks of scan data to create signatures for images in the PNG format. While the signatures in this work are not unique, the extraction time is much faster than full file hashing. The authors suggest that inexpensive and less accurate techniques may be used to rule out the majority of non-contraband, while more expensive methods may be used to verify potential contraband. This allows for reduced processing loads and efficient contraband detection.

3.2 Exploiting JPEG Header Features

Cryptographic hashing is usually applied in a manner which is indifferent to the specifics of the file format, processing the entire file at the binary level. However, there is much to be gained through careful analysis of the JPEG file structure, with the following work focusing on the utility of features found in the JPEG header.

As JPEGs are often a central part of many investigations, it is important that their integrity can be verified, so as to detect manipulation or forgeries. Piva (2013) provides an overview of techniques addressing both of these issues. Signal processing techniques can be used to identify imperfections or camera fingerprints caused by particular camera lenses; image sensors; and camera software traces, such as colour filters. Image manipulations are detected both in the JPEG pixel and compression domains, using 8×8 block boundaries, pixel and frequency domain gradients, and coefficient histograms.

In contrast, a body of work has developed which attempts to address the issues of integrity and source identification purely via the exploitation of JPEG headers. Quantization table fingerprinting has been explored for this purpose (Farid, 2006, 2008; J. D. Kornblum, 2008; Mahdian, Saic, & Nedbal, 2010), with findings suggesting that while quantization tables are not unique, they provide a good deal of discriminating information. Quantization tables may be unique to a particular camera model/software editor, or they may be able to identify a particular manufacturer, or group of source devices (Farid, 2006, 2008). Together with knowledge of the base tables provided in the JPEG standard, it is possible to detect mismatches and identify manipulation (J. D. Kornblum, 2008). However, this approach fails when adaptive quantization tables are used, which adjust the table on a per image basis, much in the same way that Huffman tables may be optimised (J. D. Kornblum, 2008).

Gloe (2012) provides an analysis of quantization tables as well as structural information pertaining to metadata, file markers and thumbnails. It was noted that the ordering and particular structure of metadata may be used as a mechanism to detect image manipulation. This is attributed to differences between how image editors record this data at the lowest level, essentially leaving a software fingerprint behind which can be detected, hindering untraceable modifications. It is suggested that convincing forgeries require advanced programming skills in order to avoid altering these structures using existing tools.

Kee, Johnson, and Farid (2011) utilise a larger set of JPEG header features to generate signatures for cameras and software tools. In addition to quantization tables, features are extracted from Huffman tables, EXIF metadata, image dimensions and thumbnails. This was shown to be effective for 1.3 million Flickr images, with 62% of signatures identifying a single camera model, 80% to three or four cameras, and 99% identifying a unique manufacturer. The algorithm does not use complete representations of the original data structures and instead uses only the number of Huffman codes of each length, and simple counts of EXIF fields. From the extracted features, EXIF data was shown to be the most distinct, followed by the image dimensions.

In the domain of Content Based Image Retrieval (CBIR), Edmundson and Schaefer (Schaefer, Edmundson, Takada, Tsuruta, & Sakurai, 2012; Schaefer, Edmundson, & Sakurai, 2013; Edmundson & Schaefer, 2012, 2013) exploit optimised Huffman tables for inexpensive image comparison. This is possible as optimised tables are derived from the frequencies of

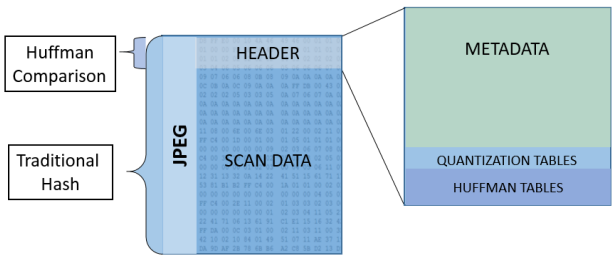


Figure 2: **The portions of a JPEG used for traditional cryptographic hashing and Huffman comparison.**

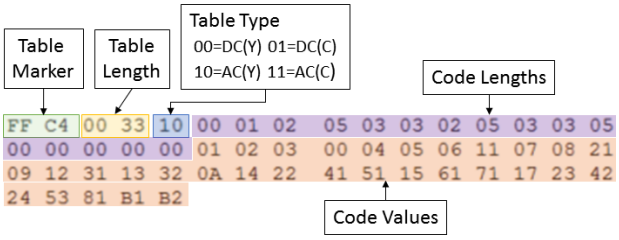


Figure 3: **An example Huffman table as it appears on disk. In table types, Y corresponds to the luminance channel, while C corresponds to the chrominance channels (Cb/Cr)**

DCT coefficients in the compressed data stream, and can therefore serve as a coarse proxy for image content. Image search performance was evaluated using a number of datasets, including Flickr 1 million (Huiskes et al., 2010), with performance levels being comparable to prior CBIR methods, while offering a 30 fold computational improvement over non-Huffman based compressed domain methods, and 150-fold over the fastest pixel domain method.

4. METHODOLOGY

To generate ranked similarity lists and to detect image source devices and software, prior work utilised entire Huffman tables. Our approach is to use optimised Huffman tables to identify particular JPEG images. This means that only the header of the JPEG needs to be read, while traditional hash based fingerprinting processes the entire file, as depicted in Figure 2.

4.1 Extracting Huffman tables

Structures in the JPEG format are preceded by markers which consist of two bytes, which always begin with `0xFF`, with the second byte indicating the type of marker.

An example Huffman table is provided in Figure 3. Huffman tables are essentially stored as two arrays, the first containing the number of Huffman codes of each bit length, while the latter lists the corresponding DC and AC byte codes in the table. These arrays are all that is needed to reconstruct the entire Huffman decode tree. Pseudo-code for generating Huffman fingerprints is provided in Algorithm 1, showing that an ordered concatenation of all length/value arrays is all that is required to generate the fingerprint.

In practice, there are many existing libraries for parsing JPEGs. The authors chose the Libjpeg (Independent JPEG Group, 2016) library to extract the Huffman tables using C++. Once the de-compression object is initialised, header data, including the Huffman tables, can be acquired by calling the `jpeg_read_header` function. This header information also includes JPEG type and colour space data. At this point, the file has been processed up to the Start of Scan (SOS) marker, which indicates the beginning of the compressed data streams.

4.2 Properties and Limitations

Optimised Huffman tables provide coarse information about the relative frequencies of DCT codes in the compressed data stream, and therefore communicate some information about the frequency domain representation of the source image. As such, fingerprints derived from the DCT codes are robust to metadata modification and some modifications to image content.

A limitation of this technique is that both images must have been encoded using the same quantization tables, colour space, and channel sub-sampling. If this is not the case the images will contain a different distribution of DCT byte codes, resulting in different Huffman tables when optimised.

Progressive JPEGs use multiple scans at different resolutions, potentially resulting in many more Huffman tables than baseline JPEGs. As

Algorithm 1: Generate Huffman Fingerprint. Order tables by type to avoid issue with ordering on disk.

```
Input: JPEG File
Output: JPEG Fingerprint
huffmanTables= {};
marker = nextMarker();
// Loop until SOS marker
while (marker != 0xFFDA) do
    // Check for Huffman marker
    if (marker == 0xFFC4) then
        length = readBytes(2);
        type = readBytes(1);
        // read remaining length after
          length/type bytes
        htable = readBytes(length-3);
        huffmanTables[type]=toString(htable);
    end
    marker = nextMarker();
end
// Order keys by type: 0x00, 0x01, 0x10,
  0x11
orderedKeys = order(huffmanTable.keys());
fingerprint = ' ';
for (key in orderedKeys) do
    fingerprint += huffmanTable[key];
end
return fingerprint
```

such, encoding the same image as both baseline and progressive JPEG will result in a mismatched number of tables, though the tables from the baseline image may be very similar to the tables for coarse progressive scans. Huffman tables will appear before each scan in progressive JPEG, such that they will be found throughout the file.

The process of generating optimised Huffman tables involves inspecting the corresponding DC and AC streams, and counting DCT code frequencies. The most frequently occurring items are assigned the smallest Huffman codes. The `jpegtran` utility in the Libjpeg package may be used to create optimised images from unoptimised JPEGs with the `-optimize` option. However, this requires the entire file to be processed, and introduces substantial overhead. Therefore, while it is possible to acquire optimised tables for any given image, Huffman table comparison is best suited to images which are already optimised.

4.3 Evaluation

In order to demonstrate the utility of optimised Huffman table analysis for digital forensics, several features must be investigated. The first is the distinctness of optimised Huffman tables, and whether they are capable of identifying uniquely images within a large dataset. Secondly, the incidence of optimised Huffman tables in the wild must be explored, in order to determine how often this technique can be applied. The proportion of a JPEG required to read the Huffman tables, and any corresponding reduction in processing time, must be also be quantified to measure the performance advantage of this technique.

Huffman fingerprints are constructed by extracting the Huffman data, ordering the data by table type, and concatenating the raw data. For the purposes of the experiment, where two fingerprints match, clashes were verified using a SHA256 digests of the entire images.

The offset of the Start of Scan marker was acquired after reading the JPEG header by calling the `stdio::ftell` function and then subtracting the remaining bytes in the `src` input buffer to get the correct value.

Benchmarks compare the proposed method against a traditional full file hashing method using the SHA256 algorithm. Benchmark times correspond to the duration for extracting signatures from a list of files, without storing the signatures or performing database lookups. In order to assess the IO costs of accessing small pieces of the file from the storage media, an additional benchmark was performed which read the first 4KiB of each file without any processing.

4.4 Datasets

Three datasets were used in this work. The first, the Flickr 1 Million dataset (Huiskes et al., 2010), contains 1 million JPEGs with op-

timised Huffman tables. The second is the Govdocs JPEG corpus (Garfinkel et al., 2009), which contains approximately 109,000 JPEGs possessing mixed properties. The third dataset is a copy of Govdocs where all images were optimised using using JPEGtran with the `-copy all` and `-optimize` flags. Duplicate images were not removed.

While the Flickr 1 Million dataset is almost completely comprised of optimised JPEGs, three images (621224.jpg, 636616.jpg, 646672.jpg) were found to use default Huffman tables, and therefore produced the same Huffman fingerprint. These three images were optimised using the same method as for the optimised Govdocs dataset. All optimised images use the `YCbCr` colour space and are non-progressive, while the unmodified Govdocs dataset contains mixed types.

5. FINDINGS

5.1 Huffman Distinctness in Flickr 1 Million

If Huffman tables are to be used to identify particular images in a large dataset, they must contain enough discriminating power to do so. To this end, the Huffman fingerprints for all images in the Flickr 1 Million dataset were used to construct equivalence classes, grouping together images with the same fingerprint. A class size of n indicates that n images possess the same fingerprint, with a class size of 1 indicating a unique fingerprint.

The Flickr 1 Million dataset contains many sets of duplicates, with a total of 746 images having at least one other image in the dataset with identical binary data (see Table 1). Once duplicates were removed from the results **all but two pairs of images were found to possess unique Huffman fingerprints**. This shows that this fast fingerprinting technique has almost a zero percent false positive rate at scale.

Indeed, the two sets of JPEGs with matching Huffman tables in this collection are almost identical, and in reality differ by a small number of pixels. Image differences were visualised using the Resemble.js library (Cryer, 2016), with differences highlighted in Figure 4. In the first case, 3 pixels are different as a semi-colon is added to the text rendered in the image, while in the second case one version of the image has two letters transposed. As the matching pairs also use the same quantization tables, such differences are small enough to result in the same optimised Huffman tables being generated.

Figure 4: **Highlighted image differences for image pairs with matching Huffman tables in the Flickr 1 Million dataset. Images 985964.jpg and 986229.jpg are represented on the left, 431419.jpg and 431931.jpg on the right.**

This result shows that optimised Huffman tables possess a great deal of discriminating power, with only two pairs of nearly identical images possessing the same Huffman fingerprint in a dataset of 1 million images. Indeed, this may be seen as a positive property, as the fingerprint can be tolerant to slight changes within the image.

5.2 Huffman Distinctness in Govdocs

Two versions of the Govdocs dataset were used: *i)* the unaltered original dataset, and *ii)* a version where all images have had their Huffman tables optimised, and converted to baseline JPEGs, using `jpegtran`. The former is used to derive representative statistics for how common particular types of JPEG are in the wild, while the latter provides a secondary test dataset for optimised Huffman distinctness. A small number of JPEGs generated errors when optimising or extracting Huffman tables and other information, and those were omitted from this analysis.

The unaltered Govdocs corpus is heterogenous, with a mix of JPEG modes, colour spaces

Flickr 1 Million	Equivalence Classes				
Size	**1 (Unique)**	**2**	**3**	**4**	**5**
No. Images Huffman	999250	726	18	4	5
No. Images Sha256 Duplicates	N/A	722	15	4	5
No. Images Huffman No Dupes.	999250	4	0	0	0

Table 1: The number of images belonging to equivalence classes of each size for the Flickr 1 Million dataset. A class size of n indicates that there are n images with the same fingerprint.

Colour Space	YCbCr	YCCK	CMYK	RGB	Greyscale
No. Images	95783	3	0	9	13443

Table 2: The number of images for each colour space option in the Govdocs corpus.

and origin software. Of approximately 109,000 images, only 6809 JPEGs (6.2%) use the progressive format, with the remainder using the baseline JPEG format. 37,879 (34.7%) baseline JPEGs use default Huffman tables, and, as such, produce the same fingerprint when extracted. 39,035 images (35.7%) contained Adobe application markers, which may either use optimised or pre-defined Huffman tables. The predominant colour space is overwhelmingly `YCbCr`, as shown in Table 2.

Prior to optimisation, 39,328 (36%) of all Govdocs images have a unique Huffman fingerprint, which is 55.1% when excluding default tables. The remaining images are primarily grouped in to very large classes, with 23,706 images belonging to groups of equivalent Huffman tables with 1000 or more members, the largest class containing over 10,000 images. This indicates that, even in the presence of many JPEGs using predefined tables, Huffman analysis can be used as the sole method of identifying images more than 1/3rd of the time. That is, this corpus suggests that optimised Huffman tables are used as often as default tables, however the authors argue that the trend is towards optimised JPEGs, and indeed the Govdocs corpus itself is relatively old. The software developed by Mozilla (Mozilla, 2017) and Google (Alakuijala et al., 2017) may be an indication that optimised images will appear more frequently on the Web, where page load times and data transfers are relatively expensive compared to other domains.

The optimised Govdocs dataset provided similar results to the Flickr 1 Million dataset (see Table 3), in that images with matching Huffman tables were either identical in the binary domain, or demonstrate small variations of the same image. When combining both datasets into one corpus of over 1.1 million images, no new equivalence classes were found. This confirms that Huffman tables are very distinct for optimised JPEGs, even across datasets.

5.3 Start of Scan Marker Offsets

Statistics for the position of the Start of Scan marker are depicted in Table 4 for all datasets, with a visual representation for Flickr and unmodified Govdocs in Figure 5. As the SOS marker appears after the Huffman tables, the data shows that very few 4096 byte media blocks are required to read those Huffman tables. In the case of the Flickr dataset, a single block read suffices for 96.6% of the dataset, with three blocks being sufficient for 99.6% of images. The figures are slightly higher for the Govdocs corpus, which contains more metadata, where nine blocks are required to acquire 99% of Huffman tables. In

Optimised Govdocs	Equivalence Classes				
Size	**1 (Unique)**	**2**	**3**	**4**	**5**
No. Images Huffman Only	108539	684	4	0	0
No. Images Sha256 Duplicates	N/A	676	0	0	0
No. Images Huffman No Dupes.	108539	8	4	0	0

Table 3: The number of images belonging to equivalence classes of each size for the optimised Govdocs dataset. A class size of n indicates that there are n images with the same fingerprint.

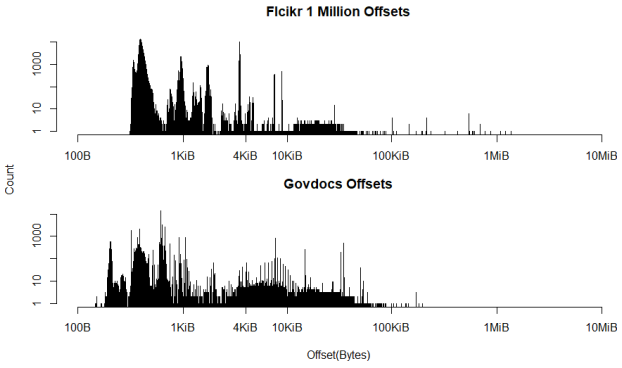


Figure 5: **Log-log distribution of Start of Scan offsets for Flickr 1 Million and unmodified Govdocs. Optimised Govdocs is almost identical to the original.**

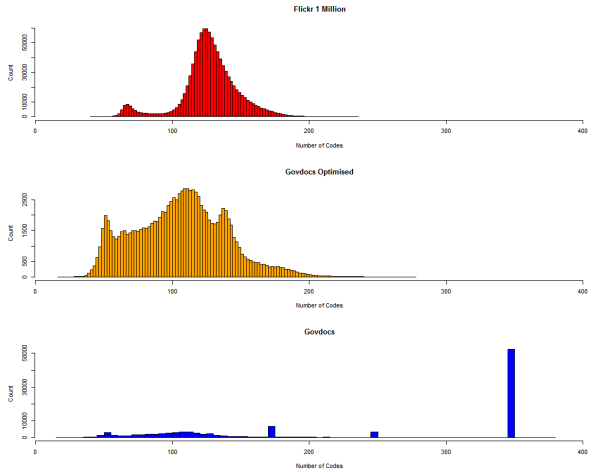


Figure 6: **Distributions for the number of DCT codes in each dataset. Flikr 1 Million the top, Optimised Govdocs in the middle, and Govdocs at the bottom.**

both cases, the distribution is long tailed, with the majority of images requiring a single block.

Using mean values for marker offsets and file lengths, 1.6% of the file must be read on average to acquire the SOS marker in the Flickr 1 Million dataset, while both Govdocs datasets require 1.2% of the file to be processed. However, when considering that 4096 bytes may be the minimal transfer size on modern storage media, the figure for the Flickr dataset rises to 3.2%, while Govdocs remains all but unchanged.

Using the proposed fast fingerprinting method, a small fraction of the file is all that is required to be read, as opposed to the entire file for traditional hashing.

5.3.1 Number of Codes and Table Lengths

The maximum number of DCT byte codes possible in the baseline JPEG format is 348 (12 per DC, 162 per AC table). However, the maximum number of codes observed for an optimised JPEG in this work was 277, suggesting that the number of codes may be used as a heuristic to distinguish between optimised and pre-defined tables. However, as can be seen in Figure 6, not all predefined tables use all codes. The spikes in the bottom graph of Figure 6, for 174 and 249 codes, are caused by images produced by Adobe Photoshop's 'save for web' settings, which optimise

Dataset	Percentile (B)					Mean (B)
	50	75	95	99	99.9	
Flickr 1 Million	973	3560	3599	9060	28866	2054
Govdocs	623	4181	23926	36128	51658	4205
Govdocs Optimised	417	3972	23863	36205	51426	4080

Table 4: Start of Scan offsets in bytes for all datasets.

entropy encoding using alternative mechanisms. However, based on this data, images with less than 300 codes are very likely to make use of optimised JPEGs.

Ignoring Huffman table markers, the length of the Huffman table may be calculated by summing the number of entries in the value and length vectors for each table. The maximum possible number of codes is 376 for baseline JPEGs, plus 16 bytes of marker and metadata for each of the four Huffman tables, for a total of 440 bytes. The maximum length found for an optimised JPEG in this work is 300 bytes, with a mean of 191 bytes for the Flickr dataset.

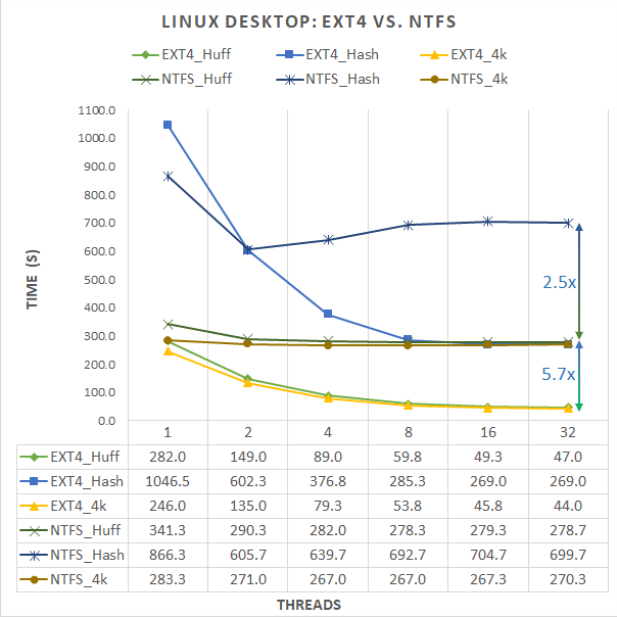
Using this observation, it is possible to identify some images with a high degree of certainty which use unoptimised tables. Additionally, this table length information indicates the number of bytes which are required to be stored for JPEG Huffman fingerprints. To reduce storage, these fingerprints themselves could be hashed using a cryptographic hashing mechanism with fixed length digests.

5.4 Benchmark Results

To quantify the potential speed improvements over traditional cryptographic hashing, several benchmarks were run on a workstation (i5-6490k, 16GiB DDR3 RAM, Western Digital Red 4TB HDD, Crucial MX300 525GB SSD) and laptop (i7-5500U, 8GiB DDR3 RAM, Samsung 840 EVO 500GB SSD), with several multi-threading options. Times represent the total extraction time for all files, with no database lookup included. Testing was limited to the Flickr 1 Million dataset, as this is both the largest dataset, and the worst case performance scenario (with the mean file size being 1/3 that of Govdocs, such that the header is a larger proportion of the file). Benchmarks were carried out on Ubuntu 15.04 64bit, with memory caches being cleared between runs. A C++ application was compiled in g++ using Boost 1.55 for thread pools, libjpeg62 for JPEG parsing, and OpenSSL for cryptographic hashing (SHA256).

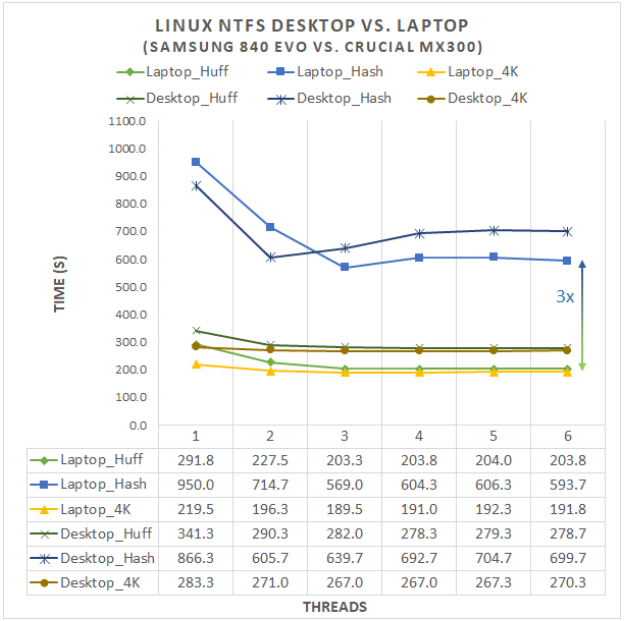
The proposed method saw no improvement when images were stored on the HDD. This can be attributed to the relatively small file sizes of this dataset, but more importantly, due to the the poor small block read performance of the mechanical media.

However, substantial performance gains were seen when utilising solid state media, which are better suited to random access patterns. Figure 7 show results for Huffman fingerprint extraction, full cryptographic file hashing, and reading the first 4096 bytes of the file with no further processing. Results are compared across both the EXT4 and NTFS file systems. While both file systems scale well to two threads, NTFS performance appears to plateau at this stage, while the EXT4 file system performance improves with the number of threads. Overheads were explored by obtaining the logical block addresses (LBA) for each file and running the experiment with physical addresses, rather than looking up each file in the file system. When the initial pre-processing was not included in the recorded time, benchmark times using the LBA addresses performed nearly identically to those of EXT4. The relatively poor performance of NTFS could be attributed to overhead within the file system, which does not scale well with many concurrent accesses. This observation was verified using the Windows operating system to rule out the Linux `ntfs-3g` driver as a bottle-



	1	2	4	8	16	32
EXT4_Huff	282.0	149.0	89.0	59.8	49.3	47.0
EXT4_Hash	1046.5	602.3	376.8	285.3	269.0	269.0
EXT4_4k	246.0	135.0	79.3	53.8	45.8	44.0
NTFS_Huff	341.3	290.3	282.0	278.3	279.3	278.7
NTFS_Hash	866.3	605.7	639.7	692.7	704.7	699.7
NTFS_4k	283.3	271.0	267.0	267.0	267.3	270.3

Figure 7: **Comparison of the relative performance of Huffman and Hash fingerprint extraction to reading the first 4K file block, across the EXT4 and NTFS file systems. EXT4 performance is close to using raw LBA block addresses.**



	1	2	3	4	5	6
Laptop_Huff	291.8	227.5	203.3	203.8	204.0	203.8
Laptop_Hash	950.0	714.7	569.0	604.3	606.3	593.7
Laptop_4K	219.5	196.3	189.5	191.0	192.3	191.8
Desktop_Huff	341.3	290.3	282.0	278.3	279.3	278.7
Desktop_Hash	866.3	605.7	639.7	692.7	704.7	699.7
Desktop_4K	283.3	271.0	267.0	267.0	267.3	270.3

Figure 8: **Comparison of the relative performance of Huffman and Hash fingerprint extraction to reading the first 4K file block, across two computers. The Laptop SSD possesses better random 4K Read performance and higher IOPS.**

neck. Thus, EXT4 and LBA based addressing are preferable when performing this kind of fractional file access.

When comparing Huffman fingerprint extraction to full file hashing on the workstation, a speed increase of **2.5**× was recorded on NTFS, and up to **5.7**× on EXT4. In both cases Huffman extraction performance mirrored 4KiB file read performance very closely, typically with less than 10% overhead. This suggests that Huffman fingerprint extraction is close to the theoretic limits of storage media access for small file fractions.

When a storage device with higher random 4KiB read performance is used, the relative performance of Huffman extraction to full file hashing also improves. This is depicted in Figure 8, which compares the same methods using the benchmark workstation and laptop machines, which each possess different SSDs. The laptop SSD has higher small block throughput with low queue depths, which is of benefit to both Huffman and hashing methods, despite the less performant CPU. In this case, Huffman extraction on the NTFS file system is **3**×.

Partial file access performance with HDDs are dominated by seek time, with transfer time being less of a factor. However as file sizes increase the relative performance would be expected to improve (McKeown et al., 2017), both for mechanical media and solid state devices. SSDs have much smaller effective seek times, and thus the proposed technique holds significantly more appeal for flash media. In modern systems flash media are becoming increasingly common, particularly in the laptop arena, in addition to already dominating the mobile device market. The observation that partial file access scales well on this type of media opens the door for flash storage optimised approaches to digital forensics, which may well be the dominant storage technology for personal computers in the near future.

6. CONCLUSIONS AND FUTURE WORK

This paper has explored the potential to use optimised Huffman tables to identify particular JPEGs in a collection, with tables being essentially unique across 1.1 million JPEG images with optimised Huffman tables. The Huffman fingerprint was shown to group very similar images together, while also being inexpensive to extract, resulting in speed increases of up to 5.7× on solid state media, despite the relatively small file sizes of the test dataset. On datasets of higher resolution images, this technique may be expected to perform well over an order of magnitude faster than file hashing, in line with prior benchmarks on similar fingerprint generation from PNG files (McKeown et al., 2017). This method also has the benefit of being usable for partially carved files, where only the header exists, potentially providing strong supplementary evidence in the absence of direct evidence.

The limitations are that this approach relies on Huffman tables being optimised ahead of time, and therefore excludes many camera images without initial pre-processing. However, the recent development of software to optimise JPEGs, in lieu of adopting a new compression standard, is promising and may indicate wider adoption of optimisation in the future. Additionally, it may be possible to exploit similar content derived features in future generations of image codecs, or for other file formats, such as compressed video.

Future work should explore the possibility of using the same technique on progressive JPEGs and other compressed media types. Additionally, leveraging work in the field of Content-Based Image Retrieval, Huffman fingerprints may be used to detect similar images in a digital forensics context. Fractional file processing may be explored in other scenarios, such as for de-duplication, or for processing remote/networked devices, which will have different performance characteristics which may facilitate greater performance improvements. Finally, flash optimised digital forensics techniques may be explored, which may be of great benefit to the field in the future.

7. ACKNOWLEDGEMENTS

This research was supported by a scholarship provided by Peter KK Lee.

REFERENCES

Alakuijala, J., Obryk, R., Stoliarchuk, O., Szabadka, Z., Vandevenne, L., & Wassenberg, J. (2017). Guetzli: Perceptually Guided JPEG Encoder. *arXiv preprint arXiv:1703.04421*. Retrieved 2017-05-31, from `https://arxiv.org/abs/1703.04421`

Beebe, N., & Clark, J. (2005). Dealing with Terabyte Data Sets in Digital Investigations. In M. Pollitt & S. Shenoi (Eds.), *Advances in Digital Forensics* (Vol. 194, pp. 3–16). Springer US. Retrieved from `http://dx.doi.org/10.1007/0-387-31163-7_1`

Breitinger, F., Liu, H., Winter, C., Baier, H., Rybalchenko, A., & Steinebach, M. (2013). Towards a process model for hash functions in digital forensics. In *International Conference on Digital Forensics and Cyber Crime* (pp. 170–186). Springer.

Cryer, J. (2016). *Resemble.js: Image analysis and comparison.* Retrieved 2017-02-21, from `https://github.com/Huddle/Resemble.js`

Edmundson, D., & Schaefer, G. (2012). Fast JPEG image retrieval using optimised Huffman tables. In *Pattern Recognition (ICPR), 2012 21st International Conference on* (pp. 3188–3191). IEEE. Retrieved 2016-03-16, from `http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6460842`

Edmundson, D., & Schaefer, G. (2013, November). Very Fast Image Retrieval Based on JPEG Huffman Tables. In *2013 2nd IAPR Asian Conference on Pattern Recognition (ACPR)* (pp. 29–33). doi: 10.1109/ACPR.2013.18

Farid, H. (2006). *Digital image ballistics from JPEG quantization* (Tech. Rep.). Technical Report TR2006-583, Department of Computer Science, Dartmouth College.

Farid, H. (2008). Digital image ballistics from JPEG quantization: A followup study. *Department of Computer Science, Dartmouth College, Tech. Rep. TR2008-638*. Retrieved 2016-05-06, from http://www.cs.dartmouth.edu/farid/downloads/publications/tr08.pdf

Garfinkel, S., Farrell, P., Roussev, V., & Dinolt, G. (2009, September). Bringing science to digital forensics with standardized forensic corpora. *Digital Investigation*, *6*, S2–S11. Retrieved 2016-03-05, from http://linkinghub.elsevier.com/retrieve/pii/S1742287609000346 doi: 10.1016/j.diin.2009.06.016

Garfinkel, S., Nelson, A., White, D., & Roussev, V. (2010, August). Using purpose-built functions and block hashes to enable small block and sub-file forensics. *Digital Investigation*, *7*, S13–S23. Retrieved 2016-03-03, from http://linkinghub.elsevier.com/retrieve/pii/S1742287610000307 doi: 10.1016/j.diin.2010.05.003

Gloe, T. (2012). Forensic analysis of ordered data structures on the example of JPEG files. In *Information Forensics and Security (WIFS), 2012 IEEE International Workshop on* (pp. 139–144). IEEE. Retrieved 2016-04-29, from http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6412639

Huiskes, M. J., Thomee, B., & Lew, M. S. (2010). New trends and ideas in visual concept detection: the MIR flickr retrieval evaluation initiative. In *Proceedings of the international conference on Multimedia information retrieval* (pp. 527–536). ACM. Retrieved 2017-02-22, from http://dl.acm.org/citation.cfm?id=1743475

Independent JPEG Group. (2016). *Libjpeg*. Retrieved 2017-02-20, from http://www.ijg.org/

Kee, E., Johnson, M. K., & Farid, H. (2011). Digital image authentication from JPEG headers. *Information Forensics and Security, IEEE Transactions on*, *6*(3), 1066–1075. Retrieved 2016-04-05, from http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5732683

Kornblum, J. (2006, September). Identifying almost identical files using context triggered piecewise hashing. *Digital Investigation*, *3, Supplement*, 91–97. Retrieved 2016-03-04, from http://www.sciencedirect.com/science/article/pii/S1742287606000764 doi: 10.1016/j.diin.2006.06.015

Kornblum, J. D. (2008, September). Using JPEG quantization tables to identify imagery processed by software. *Digital Investigation*, *5*, S21–S25. Retrieved 2016-04-14, from http://linkinghub.elsevier.com/retrieve/pii/S1742287608000285 doi: 10.1016/j.diin.2008.05.004

Mahdian, B., Saic, S., & Nedbal, R. (2010). JPEG quantization tables forensics: a statistical approach. In *Computational Forensics* (pp. 150–159). Springer.

McKeown, S., Russell, G., & Leimich, P. (2017). Fast Filtering of Known PNG Files Using Early File Features. In *Annual ADFSL Conference on Digital Forensics, Security and Law*. Daytona Beach, Florida, USA.

Mozilla. (2017). *Mozjpeg: Improved JPEG encoder.* Retrieved 2017-02-20, from https://github.com/mozilla/mozjpeg

Piva, A. (2013). An Overview on Image Forensics. *ISRN Signal Processing*, *2013*, 1–22. Retrieved 2016-05-06, from http://www.hindawi.com/journals/isrn.signal.processing/2013/496701/ doi: 10.1155/2013/496701

Quick, D., & Choo, K.-K. R. (2014, December). Impacts of increasing volume of digital forensic data: A survey and future research challenges. *Digital Investigation*, *11*(4), 273–294. Retrieved 2015-10-06, from http://linkinghub.elsevier.com/retrieve/pii/S1742287614001066 doi: 10.1016/j.diin.2014.09.002

Roussev, V. (2010). Data fingerprinting with similarity digests. In *Advances in digital forensics vi* (pp. 207–226). Springer. Retrieved 2016-03-04, from

http://link.springer.com/chapter/10.1007/978-3-642-15506-2_15

Schaefer, G., Edmundson, D., & Sakurai, Y. (2013, December). Fast JPEG Image Retrieval Based on AC Huffman Tables. In (pp. 26–30). IEEE. Retrieved 2016-04-15, from http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6727165 doi: 10.1109/SITIS.2013.16

Schaefer, G., Edmundson, D., Takada, K., Tsuruta, S., & Sakurai, Y. (2012, November). Effective and Efficient Filtering of Retrieved Images Based on JPEG Header Information. In (pp. 644–649). IEEE. Retrieved 2016-03-31, from http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6395154 doi: 10.1109/SITIS.2012.97

Wallace, G. K. (1992). The JPEG still picture compression standard. *Consumer Electronics, IEEE Transactions on*, *38*(1), xviii–xxxiv. Retrieved 2016-03-11, from http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=125072