Annual ADFSL Conference on Digital Forensics, Security and Law 2017

May 15th, 3:15 PM

Fast Filtering of Known PNG Files Using Early File Features

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

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

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

FAST FILTERING OF KNOWN PNG FILES USING EARLY FILE FEATURES

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 media, comparing individual digests with a database of known contraband. However, the large capacities of modern storage media, and increased time pressure on forensics examiners, necessitates that more efficient processing mechanisms be developed. This work describes a technique for creating signatures for images of the PNG format which only requires a tiny fraction of the file to effectively distinguish between a large number of images. Highly distinct, and compact, such analysis lays the foundation for future work in fast forensics filtering using subsets of evidential data.

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

1. INTRODUCTION

Digital forensics investigators have been under pressure to cope with increasingly large volumes of data for the past decade (Beebe & Clark, 2005). Despite the attention this problem has received during this period, there are still several research gaps in data mining, triage, and data reduction techniques (Quick & Choo, 2014). Many police organisations find themselves with several months of backlogs which have impeded the course of justice to the extent that the problem is now public knowledge (Goldberg, 2015).

One aspect of an investigation is the processing of large numbers of images on a given device. An existing approach for large image sets is to compute cryptographic hashes for images, or data blocks constituting an image file, which are then compared to a database of known file hashes (Garfinkel, Nelson, White, & Roussev, 2010). However, such an approach typically relies on all blocks in the file being read and processed, and detection may be foiled by small manipulations to the file. Simply changing embedded metadata, such as the title or date information, or padding existing values with zeroes, is enough to completely change the output of a cryptographic hash function. Additionally, such modifications need not impact the actual content of the file, resulting in no modification of the actual contraband content.

This work introduces a technique for filtering known images of the Portable Net-

work Graphics (PNG) based on signatures created from approximately 1% of the file. Features are extracted from early portions of the file, while only focusing on elements which are used to render the image. This approach allows for a highly distinct signature to be created which reduces the quantity of data to be processed by 99%. The PNG format is used on over 70% of websites, almost as frequently as the JPEG format (w3techs, 2017), making it a good target for forensics processing efforts.

The main contribution of this paper is three-fold. The first is a breakdown of the features of the PNG format which appear early in the file. The second is an evaluation of the discriminative power of PNG header and small data block features, with a view to creating fingerprints for unique objects. Finally, this research demonstrates the potential for highly accurate, and efficient, file signature creation which may be used as part of a pre-filtering scheme to reduce investigation processing times, with a note on fundamental limitations due to the underlying storage technology. Experiments are carried out using a dataset of approximately 100k images, which were converted to PNG from the Govdocs JPEG corpus (Garfinkel, Farrell, Roussev, & Dinolt, 2009), and a smaller, heterogeneous dataset of approximately 6500 PNGs collected from the Web.

The remainder of this paper is structured as follows: Section 2 outlines the relevant features of the PNG format, before discussing relevant literature in digital forensics and image spam detection in Section 3. Details pertaining to the test dataset and feature extraction methodology are provided in Section 4, with the evaluation of features in Section 5 and discussion in Section 6 Finally, Section 7 concludes and provides suggestions for future work.

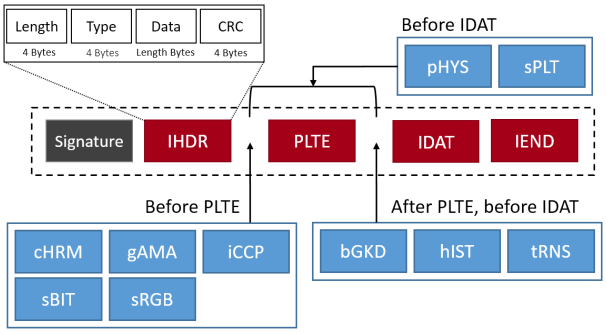

Figure 1. **The layout of a PNG file. Critical chunks in red (within dotted lines), ancillary chunks in blue, with textual and timestamp chunks omitted for clarity.**

2. THE PNG FORMAT

The PNG format is the most popular lossless compression format for images, allowing for smaller file sizes than the equivalent Bitmap Image (BMP).

The general layout of a PNG file is shown in Figure 1. PNG files begin with an 8 byte signature, containing the ASCII bytes for 'PNG', as well as various line ending and transmission integrity bytes. All file content thereafter, both metadata and compressed pixel data, is stored in 'chunks'.

Chunks are self-contained, storing their own length, data, and a checksum to detect corrupted data. Four letter ASCII labels are provided to easily distinguish between chunk types, which are referred to throughout this paper.

Labels beginning with upper case characters, depicted in red in Figure 1, refer to 'critical' chunks, which form the minimal set which an encoder and decoder should support. 'Ancillary' chunks, depicted as blue, need not be supported, and may be safely ignored, or omitted, without preventing the image from being rendered. However, such omissions mean that the image may not appear as originally intended, potentially

displaying without transparency and backgrounds, or rendering with a skewed colour spectrum.

There are 18 chunk types defined in the international standard (Adler et al., 2003), with 4 critical chunks and 14 ancillary chunks. The critical chunks are: `IHDR`, containing header metadata, which is required immediately after the signature; `IEND`, required as the final chunk to complete the file; `IDAT`, one or more of which store compressed pixel data; and `PLTE`, which stores the colour palette and is only required for the indexed colour mode.

Ancillary chunks in the specification correspond to colour space information (`cHRM`, `gAMA`, `iCCP`, `sBIT`, `sRGB`), pixel dimensions and aspect ratio (`pHYs`), suggested palette (`sPLT`), miscellaneous information for the suggested background (`bKGD`), colour histogram (`hIST`), transparency information (`tRNS`), textual information (`iTXt`, `tEXt`, `zTXt`), and timestamp information (`tIME`). Additionally, custom chunk types can be included by the encoder for use with specific applications, as with Adobe Photoshop's proprietary chunks. With the noted exception of the beginning and end chunks, there are only loose constraints placed on the order or location of chunks in the file. Most chunks are required to be present before the first `IDAT` chunk (see Figure 1), with the exception of textual information, which may be present at any point in the file prior to `IEND`.

There are five colour modes supported by PNG, which include greyscale and truecolour, both with additional modes which include an alpha channel, and the indexed colour mode. Colour modes are set in the `IHDR`, with optional reference colour points (`cHRM`) and profiles (`ICCP`) in other chunks. Transparency is supported either by means of the alpha channels or by preselected colours which indicate transparency.

Compression of pixel data is achieved by per scan line prediction and the DEFLATE compression standard, which is stored in the zlib format (Deutsch & Gailly, 1996). This data is contained in one or more `IDAT` chunks in the PNG.

3. BACKGROUND

This work is relevant to several areas of existing work. First, forensic file hashing is discussed in the context of detecting the presence of known pieces of contraband. Subsequent discussion pertains to existing approaches in digital image forensics, with a focus on image metadata, before considering similar approaches in the field of email image spam detection.

Cryptographic hashes are used both to verify the integrity of evidence and to identify known files. Kornblum (2006) noted that such hashes, which are typically based on the entire content of a file or storage media, can easily be attacked by modifying a single bit in the original data, which produces an entirely different hash digest. In order to mitigate this, piecewise hashes may be used, which increase the hash resolution by individually hashing subsets of the data. Kornblum (2006) extends this idea by applying a rolling hash system which enables a conservative estimate of the number of identical bytes in a file, providing a similarity score from 0–100. Techniques are required for fast comparisons of the large number of hashes stored in known file databases, with bloom filters commonly being utilised as an efficient lookup datastructure (Roussev, Chen, Bourg, & Richard, 2006; Roussev, 2010; Roussev, Quates, & Martell, 2013; Penrose, Buchanan, & Macfarlane, 2015).

Piecewise hashing can give false positives due to the existence of common data blocks in various file types. Roussev (2010) describes a method for selecting statistically

improbable features when producing data fingerprints, while Garfinkel et al. (2010) focus on reducing the number of non-distinct blocks from the hash database. This authors show that, in some cases, utilising all of the information in the file may be counterproductive when attempting to create robust fingerprints.

File manipulation attacks may be executed at multiple levels. Rather than simply attempting to deceive binary level fingerprinting techniques, attacks may focus on the semantic level, with existing work showing that content preserving manipulations can be effective against content based matching schemes (Gloe, Kirchner, Winkler, & Bohme, 2007).

A substantial body of literature has developed around addressing forensically relevant questions using pixel data and metadata, and is primarily concerned with three problems: source identification, tamper detection, and identifying synthetically generated images (Sencar & Memon, 2008). Metadata is often easier to manipulate, with most solutions focusing on detecting noise patterns and imperfections introduced by camera lenses, colour filter arrays, and sensors (Chen, Fridrich, Goljan, & Luks, 2008; Sencar & Memon, 2008; Piva, 2013). However, such methods are computationally expensive, and are still vulnerable to sophisticated attacks, as demonstrated by Gloe et al. (2007).

As the JPEG file format is the most commonly used image format for general use, alternative, less costly, approaches have been proposed which make use of JPEG specific features. Kornblum (2008) uses JPEG quantization table signatures to detect images which have been edited with software. This is possible as cameras and image editing software often use distinct tables, such that the presence of an Adobe Photoshop quantization table in a photograph indicates that it has been edited. Extending this idea, Gloe (2012) utilises both quantization tables and EXIF metadata structures created by software editing tools. In this case, an analysis of metadata structures shows that different tools generate different EXIF fingerprints for the same image, which can be exploited as an additional feature when detecting compromised images. Further to this, Gloe, Kirchner, and Riess (2013) showed that JPEG metadata may survive conversion to the PNG format. This metadata was then used to identify if the original JPEG was manipulated with 100% accuracy, largely through textual metadata containing JPEG sampling factors. For the purpose of identifying source cameras, Kee, Johnson, and Farid (2011) construct fingerprints using more JPEG header information, generating features from Huffman encoding tables, image thumbnails, and image dimensions, in addition to quantization tables and EXIF metadata. The authors show that the discriminating power of any given feature is not particularly high, however the combination thereof produces signatures which are unique to a camera 62% of the time, and unique to a manufacturer 99% of the time.

Image header features have also been used to detect spam emails where the spam content has been moved to an image attachment, bypassing textual filters. Krasser, Tang, Gould, Alperovitch, and Judge (2007) make use of a subset of image metadata, deriving features from image dimensions, aspect ratio, file type, file size, image area and compression factor. These features are used to train both Support Vector Machine and decision tree classifiers, intended to be used as an initial filtering stage in the detection process. Dredze, Gevaryahu, and Elias-Bachrach (2007) make use of similar features, but include additional metadata features, such as bit depth and EXIF data, as well as more expensive edge and colour in-

formation when training classifiers. However, metadata features were found to be the most useful, particularly when optimising for speed with dynamic feature selection. Uemura and Tabata (2008) design a two stage process, incorporating similar metadata features into a traditional Bayesian filter model, with text analysis as the first stage, and image metadata the second. Liu, Tsao, and Lee (2010) utilise image header information as part of a triple layer system, where the first analyses email headers data, the second examines image header data, with the third stage performing costly pixel domain analyses.

4. METHODOLOGY

This section describes the early file features of the PNG format which are used to create signatures for the purposes of identifying known images. This is in contrast to prior work which uses file header and metadata information for the purposes of classifying images in terms of source, spam content, or manipulation status.

4.1 Dataset

A dataset of 108,885 PNG images was generated by converting the original Govdocs JPEG corpus (Garfinkel et al., 2009) to PNG using the Python Pillow 3.1.1 (Clark, 2016) library. Duplicate images, determined using the `SHA256` algorithm (Gallagher & Director, 2008), three images which failed to convert, and a single corrupt PNG, were discarded. The PNG dataset (148 GiB) is larger than its original JPEG counterpart (35 GiB), with file sizes ranging from 170 bytes to 38.2MiB, with a median of 344KiB and a mean of 1.4MiB. References to the Govdocs dataset throughout this work refer to the transformed PNG version.

As the Govdocs conversion was achieved by processing all images with the same software, and may therefore contain homogeneous properties, a supplementary dataset of 6469 (2.36 GiB) images was acquired by issuing varied queries to the Bing search engine, with results being collected from 2750 unique top level domains. These files were generally smaller, with a similar range from 189 bytes to 15.2MiB, median of 132KiB and a mean of 381KiB.

4.2 Feature Extraction

Feature extraction focuses on early parts of the file, such as the file header and the first `IDAT` data chunk, for two reasons. The first is that forensic file recovery by means of data carving, which typically leverages file type signatures, often results in partial files, such as the file header and contiguous, unfragmented, data blocks. Secondly, a reduction in the portion of the file required for identification also means reduced disk access overhead, which is generally the limiting factor in digital forensics (Richard & Roussev, 2006), allowing for fast image processing.

4.2.1 Header Features

The initial intention was to leverage specific features with potentially high discriminating power, such as colour histograms (`hIST`), colour palettes (`PLTE`) and low resolution image scans from interlaced PNGs (early `IDAT`). However, none of these features were used frequently in the heterogeneous Bing dataset (7.2% paletted images and 2.9% interlaced, and no instances of the histogram chunk), such that a more general approach was taken.

The PyPNG (Jones, n.d.) python module was used to extract features from PNG chunks prior to the first `IDAT` chunk. This was achieved by using the *Preamble* method of the PNG *Reader* class, which processes image metadata from a common subset of PNG chunk types. The chunk types considered, and their associated features, are pro-

vided in Table 1. When the term 'header' is used in the remainder of this document, it is in reference to these chunks. Derivative features, such as number of planes or the alpha flag, are omitted, as their information is contained in the colour type.

Ideally, features should only be extracted from the subset of chunks which contribute to the rendering of the image, ancillary or otherwise. Such features are safe from arbitrary tampering without affecting the way the image displays. Notable omissions for the header features include those containing textual information, timestamps, ICCP colour profiles, standard RGB colour space flags, and proprietary Adobe Photoshop chunks, none of which are obtained using the above python method.

As the `IHDR` chunk is required to be in every image, it was evaluated in isolation from other metadata chunks, serving as a minimal header baseline.

To facilitate inter-image comparisons, all features were concatenated into a single string which takes into account feature order, allowing for simple string equality to determine whether two images have the same feature vector.

4.2.2 Chunk Order

Gloe (2012) noted that image encoders for the JPEG format are not necessarily consistent with the order in which they include metadata structures, such that the order of elements can be used as a discriminating feature. As noted previously with regards to chunk placement constraints (see Section 2), this is also true in the PNG specification (Adler et al., 2003). The `IHDR` and `IEND` chunks must be first and last respectively, however, other chunks have loose, or no, ordering constraints. For example, the `gAMA` and `sBIT` chunks must appear before both the `PLTE` and `IDAT` chunks, but could appear in either order.

To determine the potential effectiveness of chunk orders as a feature, all chunk types were aggregated in to an ordered list to two depths in the file: ***i)*** All chunks in the file, and ***ii)*** Chunks up to the first `IDAT` chunk. By varying how far into the file is processed, it is possible to determine how much of the file must be analysed for this feature to be useful, if at all. All chunk types, including proprietary and textual metadata chunks were included, with subsets not being considered due to the initial experimental performance.

Again, to facilitate simple equivalence comparison, and integration with other features, ordered lists are transformed into strings.

4.2.3 Cryptographic Hashes

PNG header chunk(s) are immediately followed by an `IDAT` chunk, which contains compressed image data. Even if a limited portion of the file were to be recovered, it is likely that some of this data would be included in the first disk block, except in cases where there are many metadata chunks. As such, a small portion of scan data is included as a feature by calculating `SHA256` hashes of the first n bytes of the first `IDAT` chunk, for varying sizes of n.

Additionally, traditional block based hashing, which is indifferent to the structure of the file, is also performed to serve as a baseline with which to evaluate the relative performance of all features discussed. In this case, n bytes from the beginning of the file are used to generate hashes using the same method.

When the value of n is less than, or equal to, the length of the hash digest it would produce (32 bytes), the raw data is included in the signature instead of the hash digest.

Feature	Chunk	Description
File Signature	N/A	'Magic number' from the file.
Height	IHDR	Image height in pixels.
Width	IHDR	Image width in pixels.
Bit Depth	IHDR	No. of bits per sample or palette index.
Colour Type	IHDR	Colour mode flag (See Section 2).
Filter Method	IHDR	Filter method byte, standard only uses 0 for adaptive.
Interlacing	IHDR	Image interlacing byte, 0 if no interlacing, 1 for Adam7 interlacing.
Compression Method	IHDR	Compression method byte, PNG standard only uses 0 for Deflate.
Palette	PLTE	1–256 RGB palette values if present, otherwise `None`.
Background Colour	bkGD	Default background colour, otherwise `None`.
Transparency	tRNS	Simple transparency alpha value, single colour value, or `None`.
Gamma	gAMA	Four byte floating gamma value.
Unit is Metre	pHYs	1 byte flag indicating if the unit of measurement is metre.
X-Axis Pixels per Unit	pHYs	X-axis aspect ratio or metric size information.
Y-Axis Pixels per Unit	pHYs	Y-axis aspect ratio or metric size information.
Significant bits	sBIT	Stored the original number of significant bits for lossless recovery.

Table 1. Header features and their associated PNG chunks.

4.2.4 IDAT Length

The size of `IDAT` chunks normally corresponds to the size of the memory buffer used by the encoder (Adler et al., 2003), though they need not be a constant length. As the beginning of each chunk stores its own length, only the first few bytes are required to obtain its size. This means that the length of the first `IDAT` chunk, immediately following the metadata, may be used as a proxy for the buffer size of the encoder, and as a discriminating feature.

The length of the first `IDAT` chunk was obtained by inspecting the *atchunk* property of the PyPNG*Reader* class immediately after calling the *Preamble* method. This returns a tuple, the first of which is the length of the chunk.

4.2.5 Combining Features and Equivalence Classes

As all features above can be easily represented as a string, simply concatenating individual features together, in an ordered fashion, allows for a single signature which can be compared quickly. This also allows the signatures for known files to be looked up in the same way that traditional cryptographic file hashes are utilised.

Equivalence classes are generated by recording a list of all images which have identical feature strings. The number of files in a given list corresponds to the size of the equivalence class. This provides a granular measure of how unique a given signature is and allows for the identification of groups which possess the same signature. A class size of 1 indicates that the signature is unique for all files, a class size of 2 means that 2 files possess the same signature, and so on.

4.3 Offset Acquisition

In order for the above information to be included in discriminating features, it has to be read from the storage media. As such, the closer to the beginning of the file the above features are located, the smaller the proportion of the file which has to be analysed. The

IDAT is the anchor point for all of the above features, as all header features must appear before it, and scan data appears immediately after it. As such, the byte offset of the first IDAT chunk is used to evaluate how much of the PNG file needs to be processed for these signatures to be generated.

5. FINDINGS

An evaluation of each feature is provided in isolation, before exploring the potential of combining features to form a single signature, with a comparison to the baseline hashing methods. The best case scenario for each feature is that it generates a unique signature for each file, i.e., an equivalence class of 1 for all images.

5.1 Chunk Orders

There are a wide range of possible chunk types, with the potential for individual applications to include proprietary chunks. The order in which these chunks appear may be used as a feature to discriminate between images. Table 2 shows the results of creating equivalence classes of all chunk types for the entire file.

Even utilising all chunks in the file, this feature performs poorly across both datasets, with only 17.9% unique signatures for Bing images, and 0.1% for Govdocs images. Chunk ordering in the Govdocs dataset barely has any discriminating power due to the homogeneity of how the files are constructed. In some cases JPEG metadata, such as ICCP profiles were retained during PNG conversion, but such incidences do not provide much information in the absence of additional ancillary chunks. The Bing dataset has more variety due to additional metadata blocks and encoding differences, though there are still only a small number of critical chunks included in the specification, limiting the usefulness of this feature.

The performance of chunk orders falls further if only chunks up to the first IDAT are considered, with the number of unique signatures dropping to 2.4% on the Bing dataset. These results suggest that the majority of the discriminating information is provided by the number of IDAT chunks included in the file, as well as any metadata chunks which are interleaved with scan data, or found after it.

5.2 IDAT Length

The length of the first IDAT chunk can be used to gain information about the number of potential data chunks in the PNG, as well as information about the encoder, without needing to analyse the entire file. The performance of this feature varies wildly between datasets (Table 2), with 60.4% unique signatures for the Bing dataset, but only 12.4% for the Govdocs dataset. As the Govdocs dataset was produced using a single encoder, the behaviour of this feature on this dataset essentially reflects the behaviour of the Pillow library. In this case, the encoder limits the length of IDAT chunks to 64KiB, with larger files being divided in to multiple chunks of this size or less. Unique signatures are produced by files which are smaller than this upper limit, with the distribution depicted on the right of Figure 2.

In the Bing corpus, IDAT lengths range from 104 bytes to 8.77MiB, the distribution of which is provided in the left side of Figure 2. These results indicate that many encoders opt not to limit chunk sizes, which is possible as the maximum chunk size in the specification is $2^{31} - 1$ bytes (Adler et al., 2003). Large spikes in the Bing distribution are likely caused by the use of the same encoding library, or those using similar, content independent buffer presets.

The IDAT chunk length is not a suit-

	Equivalence Class Size No. Images (% of Images)				
	1 (Unique)	2-5	6-10	11-100	>100
Bing					
All Chunks Order	1159 (17.9%)	1017 (15.7%)	329 (5.1%)	959 (14.8%)	3007 (46.5%)
IDAT Length	3907 (60.4%)	130 (2.0%)	14 (0.2%)	245 (3.8%)	2175 (33.6%)
Govdocs					
All Chunks Order	121 (0.1%)	364 (0.3%)	402 (0.4%)	4341 (4.0%)	103657 (95.2%)
IDAT Length	13501 (12.4%)	6266 (5.8%)	6 (<0.1%)	0	89118 (81.9%)

Table 2. Equivalence classes for chunk type orders and IDAT lengths for both datasets.

	Equivalence Class Size No. Images (% of Images)					
	1 (Unique)	2	3	4	5	>5
Bing						
IHDR Only	4638 (71.7%)	482 (7.4%)	153 (2.4%)	116 (1.8%)	50 (0.8%)	1032 (15.9%)
Header Features	5106 (78.9%)	444 (6.9%)	135 (2.1%)	92 (1.4%)	65 (1.0%)	629 (9.7%)
Govdocs						
IHDR Only	27059 (24.9%)	7626 (7.0%)	4143 (3.8%)	2868 (2.6%)	2230 (2.0%)	64959 (59.7%)
Header Features	27059 (24.9%)	7626 (7.0%)	4143 (3.8%)	2868 (2.6%)	2230 (2.0%)	64959 (59.7%)

Table 3. Equivalence classes for the IHDR and complete PyPNG preamble features.

IDAT Block Size (B)	Equivalence Class Size No. Images (% of Images)					
	1 (Unique)	2	3	4	5	>5
Govdocs						
8	72981 (73.4%)	13932 (14.0%)	5895 (5.9%)	3304 (3.3%)	2305 (2.3%)	10468 (9.6%)
16	107257 (98.6%)	1152 (1.0%)	252 (0.2%)	84 (<0.1%)	45 (<0.1%)	95 (<0.1%)
32	108670 (99.8%)	86 (<0.1%)	42 (<0.1%)	20 (<0.1%)	5 (<0.1%)	62 (<0.1%)
64	108671 (99.8%)	86 (<0.1%)	42 (<0.1%)	24 (<0.1%)	0	62 (<0.1%)
128	108671 (99.8%)	86 (<0.1%)	42 (<0.1%)	24 (<0.1%)	0	62 (<0.1%)
256	108671 (99.8%)	86 (<0.1%)	42 (<0.1%)	24 (<0.1%)	0	62 (<0.1%)
512	108671 (99.8%)	86 (<0.1%)	42 (<0.1%)	24 (<0.1%)	0	62 (<0.1%)
1024	108671 (99.8%)	86 (<0.1%)	42 (<0.1%)	24 (<0.1%)	0	62 (<0.1%)
2048	108671 (99.8%)	86 (<0.1%)	42 (<0.1%)	24 (<0.1%)	0	62 (<0.1%)
4096	108671 (99.8%)	86 (<0.1%)	42 (<0.1%)	24 (<0.1%)	0	62 (<0.1%)

Table 4. Equivalence classes for varying block sizes of `IDAT` chunk data SHA256 hashes for the Govdocs dataset.

able standalone feature, as it is sensitive to the encoder used; however, it performs well enough to be used in combination with other features. Additionally, this feature could be useful in future work as part of a process to identify particular encoders or applications which created a given PNG.

5.3 Header Feature Distinctness

While many of the features described in Table 1 have a relatively narrow range of values, with several being comprised of a single byte, their combination provides potential to discriminate between images.

The equivalence classes generated by using features extracted by the PyPNG preamble method are given in Table 3. Using all features in Table 1, 78.9% of the Bing dataset possessed a unique signature. Less than 10% of images were in a class of five or greater, with the largest class constituting 76 images. Using the same encoder in the transformed Govdocs dataset, header features are still unique 25% of the time, which is perhaps higher than expected. Performance does not suffer much on the Bing dataset when only the `IHDR` chunk, which must be present in every PNG, is considered, with no measurable difference being made on the Govdocs dataset.

While these features are not enough to uniquely identify images in the dataset, they are reliably found immediately after the PNG file signature, and contribute a good deal of discriminating information. As such, they prove good candidates for combination with additional features. Additionally, it should be noted that not all ancillary chunks which are provided in the specification are processed by PyPNG, and, as such, performance may be improved slightly by including these additional chunk types.

5.4 Small Block IDAT Hashing

Small portions of the scan data may be used to discriminate between images. Equivalence classes for blocks of varying sizes for the Govdocs dataset are provided in Table 4, where the block size is the number of bytes, n, which are hashed from the beginning of the first `IDAT` chunk. In some cases, the block size is larger than the size of the content of the first `IDAT` chunk, in which case the actual length of the chunk is used for the hash. However, this did not occur frequently, with only 200 instances when $n =$ 4KiB. Both datasets performed similarly, as such, Bing data is omitted here.

In this experiment, small values of n were sufficient to produce highly distinct signatures. As few as eight bytes of data produce a unique `SHA256` hash digest for 73.4% of images in the dataset, with no significant benefit beyond 32 bytes, where 99.8% of signatures were unique.

An examination of equivalence classes for the larger block sizes shows that duplicate hashes are caused by images which have large contiguous arrays of bytes with the value zero. As PNG is a lossless format, it can represent large areas of contiguous, identical, values without artefacts, both for solid colour and transparent backgrounds. In these cases, when the same background value is used, hash collisions will be caused as the value of n may be too small to reach the differentiating data in the foreground. That is, while this technique is generally effective, it may produce false positives with logo style images.

5.5 Small Block File Hashing

Rather than hashing the whole file, or data contained solely in the `IDAT` chunks, the first n bytes of the file were hashed to serve as a baseline. Table 5 shows the results of these

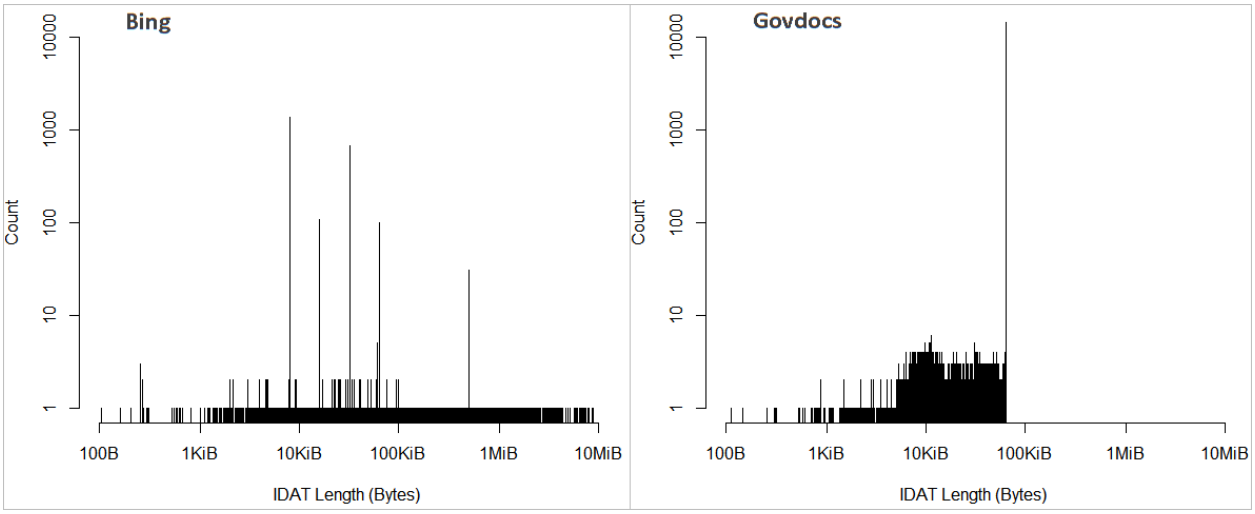

Figure 2. **The distribution of IDAT lengths for both datasets, plotted on logarithmic axes.**

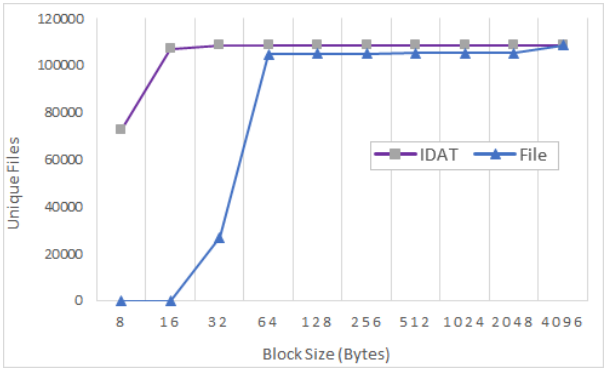

Figure 3. **The number of unique IDAT and File hashes for each block size for the Govdocs dataset.**

small block hashes. Again, performance was consistent across datasets.

Block hashes prove to be very distinct, though this strategy was not as successful as `IDAT` hashing for very small block sizes. Hashing the first 8 and 16 bytes in the file created the same signatures for all images in the dataset. When the block size reaches 32 bytes, file hashes perform identically to features extracted from the `IHDR` chunk (Table 3), as they likely contain the same content.

Hashing the first 2048 bytes of the files produced duplicate hashes more often than 16 bytes of `IDAT` data. A full disk sector (4KiB) still performs worse than 32 bytes of the first `IDAT` on Govdocs, but produces unique values for the Bing corpus. The relative performance of `IDAT` and File hashes is depicted graphically in Figure 3.

5.6 Multiple Feature Aggregation

With the exception of 4KiB file hashing on the Bing dataset, no single feature tested distinguished all distinct images. To further improve file discrimination, multiple feature strings were concatenated together to form a single signature. A graphical overview of the relative effectiveness of single and combined features for both datasets is provided in Figure 4.

As noted in Section 5.1, chunk type ordering was not a strong feature by itself, and provided minimal utility when combined with additional features. However, chunk orders can be acquired at a very low expense, while extracting other, more discriminative, features.

The best performance is achieved when

Start of File Block Size (B)	Equivalence Class Size No. Images (% of Images)					
	1 (Unique)	2	3	4	5	>5
Govdocs						
8	0	0	0	0	0	108885 (100%)
16	0	0	0	0	0	108885 (100%)
32	27059 (24.9%)	7626 (7.0%)	4143 (3.8%)	2868 (2.6%)	2230 (2.0%)	64959 (59.7%)
64	104814 (96.3%)	898 (0.8%)	408 (0.4%)	356 (0.3%)	215 (0.2%)	2194 (2.0%)
128	105034 (96.5%)	846 (0.8%)	381 (0.3%)	320 (0.3%)	175 (0.2%)	2129 (2.0%)
256	105034 (96.5%)	846 (0.8%)	381 (0.3%)	320 (0.3%)	175 (0.2%)	2129 (2.0%)
512	105447 (96.8%)	778 (0.7%)	345 (0.3%)	276 (0.3%)	160 (0.1%)	1879 (1.7%)
1024	105474 (96.9%)	762 (0.7%)	339 (0.3%)	276 (0.3%)	155 (0.1%)	1879 (1.7%)
2048	105474 (96.9%)	762 (0.7%)	339 (0.3%)	276 (0.3%)	155 (0.1%)	1879 (1.7%)
4096	108670 (99.8%)	84 (<0.1%)	42 (<0.1%)	20 (<0.1%)	0	69 (<0.1%)

Table 5. Equivalence classes for SHA256 hashes of the first n bytes of the file, ignoring file structure. Govdocs dataset.

combining small `IDAT` data block hashing with `IHDR` features and the length of the first `IDAT` chunk, with only a small number of bytes required from the scan data. This combination was able to distinguish between images in the dataset just as well as 4KiB file hashes, while providing resistance to arbitrary modifications of non-critical metadata, such as textual information.

The signatures produced are small, between 35 and 42 bytes using 16B of `IDAT` data, and between 51 and 58 bytes using 32B of `IDAT` or the `SHA256` digest.

5.7 Typical Offsets

The location of the first `IDAT` chunk in a PNG varies with the quantity of metadata included in the file. If there are few chunks included between the `IHDR` and first `IDAT` block, it can appear very early in the file, with the smallest offset in both datasets being 37 bytes. When many chunks are included, pixel data can start deep in the file, with 2.5MiB of data being present before the `IDAT` in the worst case in the Bing corpus, and 103KiB in Govdocs. However, as noted, the transformed Govdocs corpus does not contain much ancillary metadata, with the vast majority of `IDAT` blocks starting at 37 bytes.

The distribution of `IDAT` offsets in the Bing corpus is very long tailed, with a median of just 113 bytes, and mean of 4158 bytes. As the mean file size is 381KiB, the features discussed can be acquired by only reading **approximately 1%** of the file for images in the Bing dataset, with far less being required for Govdocs (0.01%). 96.9% of Bing images had an `IDAT` marker appear within the first 4096 bytes (99.9% for Govdocs), with 60% of files (92% for Govdocs) requiring a mere 160 bytes to reach the `IDAT`. With most images only requiring a single disk sector to be processed, and a very small overall proportion of the file, early file features represent a substantial reduction in disk overhead, while also producing a highly discriminative signature.

6. DISCUSSION

Based on the findings in this work, it is possible to conclude that highly distinct PNG

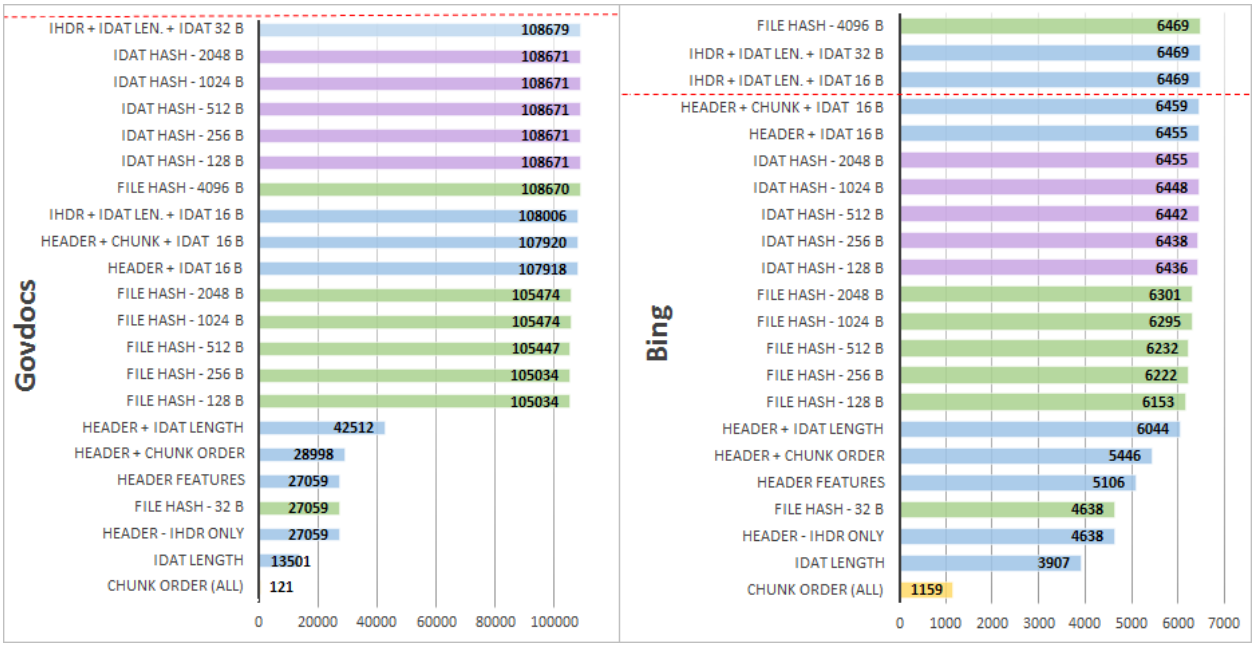

Figure 4. **The number of unique signatures for each feature and dataset. The dotted line indicates where all unique images have unique signatures.**

signatures may be derived from the mandatory `IHDR` chunk when combined with information found at the start of the first `IDAT` chunk. This contains encoding information about the image, as well as dimensions, and a proxy for the memory buffer size, with a tiny portion of scan data to rule out very similarly encoded images of the same dimensions. Using this method, a tiny fraction of the file may be used create a signature for a PNG file, while also ignoring easily stripped, non-essential, metadata modifications. Pseudocode for signature creation using these features is provided in Algorithm 1.

While this technique has been shown to be effective even on homogeneous datasets (99.8% unique signatures), it is not accurate enough to avoid false positives at scale. However, the utility of this approach lies in being able to filter out large portions of data, with potential hits being verified with more accurate, and costly, methods, such as traditional cryptographic hashing or similarity hashing. In doing so, the amount of data to be processed can be reduced by two orders of magnitude, from 100GiB to 1GiB, or less.

To quantify the potential speed improvements, several benchmarks were run on a workstation (i5-6490k, 16GiB DDR3 RAM, Western Digital Red 3TB HDD, Crucial MX300 525GB SSD) and laptop (i7-5500U, 8GiB DDR3 RAM, Samsung 840 EVO 500GB SSD), with different multi-threading options and file read orders. Benchmarks were carried out on Windows 10 64bit, with memory caches being cleared between runs. Normal file order provided by the Python *os.listdir* function, its reverse, and random orders were used. Files were copied in the same order to all drives with no fragmentation.

Benchmarks show that signatures derived from early file features were between 2–3x faster using the workstation's HDD, 8–12x faster on the workstation's SSD, and 12–18x faster on the laptop's SSD. Additionally, CPU load compared to full file hashing was reduced to less than 1/4 on the HDD and

	Order	WS_HDD Time (s)				LT_SSD Time (s)			
Threads		1	2	4	8	1	2	4	8
Header+	normal	689.6	870.9	721.0	564.9	88.2	58.6	50.9	38.6
IDAT Len.+	reverse	1129.5	1393.2	1087.5	745.9	87.1	63.7	46.0	36.0
32B	random	1463.7	2286.9	1783.9	1387.6	88.9	61.9	49.7	35.8
Full	normal	1808.1	1667.5	1649.3	1686.3	1052.6	706.2	679.9	697.2
File	reverse	2689.7	2301.6	2105.9	1859.3	1101.5	746.2	649.0	617.3
Hash	random	3192.9	2689.1	2542.8	2419.9	1111.9	734.5	632.5	591.8

Table 6. Processing times for both the proposed Signature creation and Full File Hashing for different file orders and multi-threading options. Values are averaged over multiple runs. WS and LT refer to workstation and laptop, respectively. Workstation SSD omitted for brevity.

less than 1/2 on SSDs. A summary of benchmark times is provided in Table 6.

These results suggest fundamental limitations in performance gains to be had with sub-file analyses on magnetic storage media. This can be explained by the mechanical nature of hard disk drives, which require platters to be spun in to position to meet read head actuators, which costs time to position. The heads must move physically to the next block to read, regardless of whether or not the intermediate blocks are of interest. NAND flash, on the other hand, possesses none of these mechanical limitations, instead, largely being limited by flash controller performance, communication buses and NAND layouts. Indeed, the relative performance improves when a higher throughput SSD is used. This suggests that the theoretical speed increase of two orders of magnitude (reading 1% of the file) may be realised in the future with NVMe and M.2 devices which have drastically higher random read performance and greater operations per second.

Algorithm 1: PNG Signature

```
Input: PNG File
Output: PNG Signature
signature = String(readIHDR());
while chunk.type != IDAT do
    chunk.type = nextChunk();
end
signature += String(chunk.length);
signature +=
 String(chunk.data[0:32]);
return signature
```

7. CONCLUSIONS AND FUTURE WORK

This work has shown that there are several highly discriminatory features which can be derived from PNG header chunks and a tiny portion of the first scan data chunk, in a manner which focuses on critical file information, rather than arbitrary metadata.

This approach is targeted at reducing the huge volumes of data present in a forensics investigation, providing the potential to greatly decrease the time spent reading from the forensic image on traditional storage media. This approach may not account for content manipulations when they are not re-

flected in the header; however, it serves as a method to reduce the burden of large volumes of data faced by the investigator.

Future work will investigate the relationship between image modification and header data in the PNG format, and explore variances within each chunk, which may prove useful for identifying particular software used to encode a PNG. Additional work will examine similar approaches for other forensically relevant file types. Given current hardware limitations, this kind of approach, based on partial file analysis, may be the key to streamlining, and scaling, the forensics process.

ACKNOWLEDGEMENTS

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

REFERENCES

Adler, M., Boutell, T., Bowler, J., Brunschen, C., Costello, A., Crocker, L., ... others (2003). Portable network graphics (png) specification. *Specification V1*, *2*, W3C. Retrieved from `https://www.w3.org/TR/PNG/`

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.

Chen, M., Fridrich, J., Goljan, M., & Luks, J. (2008). Determining image origin and integrity using sensor noise. *IEEE Transactions on Information Forensics and Security*, *3*(1), 74–90.

Clark, A. (2016). *Python-Pillow.* Retrieved 2017-01-07, from `https://github.com/python-pillow/Pillow`

Deutsch, P., & Gailly, J.-L. (1996). *Zlib compressed data format specification version 3.3* (Tech. Rep. No. RFC 1950).

Dredze, M., Gevaryahu, R., & Elias-Bachrach, A. (2007). Learning Fast Classifiers for Image Spam. In *CEAS.*

Gallagher, P., & Director, A. (2008). Secure hash standard (shs). *FIPS PUB*, 180–3. Retrieved from `http://csrc.nist.gov/publications/fips/fips180-3/fips180-3_final.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.

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.

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.

Gloe, T., Kirchner, M., & Riess, C. (2013, October). *How we learned to stop worrying about content and love the metadata* (Tech. Rep.).

Gloe, T., Kirchner, M., Winkler, A., & Bohme, R. (2007). Can we trust digital image forensics? In *Proceedings of the 15th international conference on Multimedia* (pp. 78–86). ACM.

Goldberg, A. (2015, November). *Child abuse cases delayed by police backlog.*

Jones, D. (n.d.). *PyPNG.* GitHub. Retrieved 2016-07-28, from `https://github.com/drj11/pypng`

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.

Kornblum, J. (2006, September). Identifying almost identical files using context triggered piecewise hashing. *Digital Investigation*, *3, Supplement*, 91–97.

Kornblum, J. (2008, September). Using JPEG quantization tables to identify imagery processed by software. *Digital Investigation*, *5*, S21–S25.

Krasser, S., Tang, Y., Gould, J., Alperovitch, D., & Judge, P. (2007). Identifying image spam based on header and file properties using C4.5 decision trees and support vector machine learning. In *Information Assurance and Security Workshop, 2007. IAW'07. IEEE SMC* (pp. 255–261). IEEE.

Liu, T.-J., Tsao, W.-L., & Lee, C.-L. (2010, August). A High Performance Image-Spam Filtering System. In (pp. 445–449). IEEE.

Penrose, P., Buchanan, W. J., & Macfarlane, R. (2015, March). Fast contraband detection in large capacity disk drives. *Digital Investigation*, *12, Supplement 1*, S22–S29.

Piva, A. (2013). An Overview on Image Forensics. *ISRN Signal Processing*, *2013*, 1–22.

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.

Richard, G. G., III, & Roussev, V. (2006, February). Next-generation Digital Forensics. *Commun. ACM*, *49*(2), 76–80.

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

Roussev, V., Chen, Y., Bourg, T., & Richard, G. G. (2006, September). md5bloom: Forensic filesystem hashing revisited. *Digital Investigation*, *3*, 82–90.

Roussev, V., Quates, C., & Martell, R. (2013, September). Real-time digital forensics and triage. *Digital Investigation*, *10*(2), 158–167.

Sencar, H. T., & Memon, N. (2008). Overview of state-of-the-art in digital image forensics. *Algorithms, Architectures and Information Systems Security*, *3*, 325–348.

Uemura, M., & Tabata, T. (2008). Design and evaluation of a Bayesian-filter-based image spam filtering method. In *Information Security and Assurance, 2008. ISA 2008. International Conference on* (pp. 46–51). IEEE.

w3techs. (2017, January). *Usage Statistics of Image File Formats for Websites, January 2017.* Retrieved 2017-01-11, from `https://w3techs.com/technologies/overview/image_format/all`

