Author's accepted version - DOI: https://doi.org/10.1109/CyberSecPODS.2018.8560680
IEEE International Conference on Cyber Security and Protection of Digital Services (Cyber Security 2018)

Sub-file Hashing Strategies for Fast Contraband Detection

Sean McKeown
School of Computing
Edinburgh Napier University
Edinburgh, Scotland
s.mckeown@napier.ac.uk

Gordon Russell
School of Computing
Edinburgh Napier University
Edinburgh, Scotland
g.russell@napier.ac.uk

Petra Leimich
School of Computing
Edinburgh Napier University
Edinburgh, Scotland
p.leimich@napier.ac.uk

***Abstract*—Traditional digital forensics processes do not scale well with the huge quantities of data present in a modern investigation, resulting in large investigative backlogs for many law enforcement agencies. Data reduction techniques are required for fast and effective digital forensics triage, and to reduce the time taken to conduct an investigation. This work explores the potential of sub-file cryptographic hashing strategies, where small fragments of files are hashed in lieu of processing the file in its entirety, for contraband detection. Results show that sub-file hashing techniques perform well, particularly on solid state media, while also retaining a high degree of discriminating power. Such strategies may offer an opportunity to take advantage of the performance characteristics of non-mechanical media, streamlining future investigations and greatly reducing investigation times.**

***Index Terms*—sub-file signatures, partial-file analysis, hashing strategies, digital forensics, cryptographic hashing**

I. Introduction

Digital data is present in an abundant volume in modern society, as mobile devices and personal computers permeate all facets of life. The pervasiveness of digital devices, and their increasing data capacities, has created a scalability problem for law enforcement agencies, which are struggling to cope with the deluge of evidence [1]. In some jurisdictions this has created delays in processing evidence of up to four years [2], potentially allowing criminals the opportunity to reoffend.

Existing solutions to tackle this data overload include random sampling, parallelisation, data mining, and machine learning [3]. However, a review [1] of these approaches has found that there are still several areas of weakness, with additional work being required to improve data reduction and triage techniques. As such, it is critical that continued research is undertaken to tackle this problem. In doing so, investigators can be supplied with more effective tools to reduce the turnaround times for evidential analysis, facilitating a reduction in existing forensics backlogs.

A. Problem Definition and Contribution

Many public sector digital forensics investigations focus on the detection and analysis of contraband images and other media. To automate this process, cryptographically secure hash signatures are used as a means of identifying exact binary copies of previously encountered contraband [4]. When analysing a new piece of evidence, such as a mobile device or hard drive, each file on the device is hashed separately and compared to a database of known contraband. When a match is discovered, the software then marks the file for the attention of the investigator.

This paper explores a data reduction approach wherein parts of files are cryptographically hashed in lieu of entire files, allowing for fast lookups of known contraband items. As the IO overhead of the storage media is typically the bottleneck in digital forensic processing, this approach offers the possibility of substantially reducing the time taken to process files on a device, aiding in the reduction of law enforcement backlogs.

The main contribution of this work is the analysis of several sub-file, file type agnostic, forensic hashing strategies, with: *i)* an analysis of the discriminating power of signatures generated by sub-file hashing strategies, and *ii)* an evaluation of the performance of sub-file hashing strategies relative to full file hashing, for both a Hard Disk Drive (HDD) and Solid State Drive (SSD) using the NTFS and EXT4 file systems.

Experiments were carried out using two datasets: Flickr 1 Million [5], containing 1 million JPEGS and the Govdocs [6] dataset, with all images converted to PNG to create a dataset with larger file sizes.

II. Related Work

A. Generating Forensic File Signatures

Data signatures are integral to the digital forensics process, being used to whitelist or blacklist previously encountered files, and to verify that the integrity of digital evidence has not been compromised. Such signatures are typically created by calculating cryptographically secure hash digests, producing signatures which are both distinct and incredibly difficult to replicate without the original data. The traditional approach is to hash every bit of data in the input file or media, however, as noted by Kornblum [7], such signatures can easily be foiled by modifying any given bit of data. A mitigation strategy may be found in piecewise hashing, dividing the input data stream into a number of chunks, calculating multiple independent hash signatures. Kornblum [7] expands on this idea by calculating a rolling hash which is used to generate cut-off points for sub-hashes. This technique, as implemented in the *ssdeep* tool, produces a similarity score which provides an estimate of the number of identical bytes when comparing a file and a signature.

The occurrence of repeating, non-distinct, blocks in a dataset may produce false positives when using piecewise hashing techniques. To combat this, Roussev [8] uses statistically improbable features when deriving data signatures, implemented in the *sdhash* tool. An alternative solution can be implemented at the block level, with Garfinkel et al. [4] choosing to evaluate the discriminative power of differently sized data blocks, generating heuristics to identify common, non-distinct, patterns which do not serve to identify a particular file.

Breitinger et al. [9] stratify signature generation approaches into: *i)* full file cryptographic hashing, *ii)* bytewise approximate matching, for approaches like *ssdeep* and *sdhash*, and *iii)* semantic approximate matching, where image signatures are derived from their visual, rather than binary, properties. These approaches were directly compared, with binary methods proving to be very fast, and effective at detecting file fragments, while semantic approaches are much more resilient to content preserving binary manipulations. The authors suggest a tiered approach for detecting contraband, based on the strength of each hashing category. Low hanging fruit is detected by full file hashing, followed by semantic approaches to detect image variants, with bytewise approximate matching conducted last to detect partial or damaged files.

A different form of signature generation is used by McKeown et al. [10], falling somewhere between bytewise approximate and semantic techniques. This approach exploits PNG encoding metadata, found in the file header, to generate signatures, combining this information with a small sample of compressed pixel data to boost the discriminating power of the technique. While this approach does not generate unique signatures for images using the same encoder, it is more resistant to binary level modifications, and greatly reduces processing times over full file hashing approaches, as only a small portion of the file is required. The authors suggest that the bulk of non-contraband can be ruled out quickly, with hits being confirmed with more expensive processing afterwards. A similar approach for JPEGs [11] makes use of optimised Huffman tables, found in the JPEG header, and used to decode image data. When JPEGs are optimised for maximal compression, these tables possess a high degree of discriminating power, producing unique signatures for the Flickr 1 Million dataset, while requiring only a small fraction of the file to be read.

Forensic signatures can be generated in many ways, and from different aspects of the file. Partial file signatures trade discriminating power for processing speed, as this is a primary concern with large quantities of data. The next section discusses techniques which are optimised for quick initial assessments of evidence.

B. Digital Forensic Triage

Digital forensic triage can be performed to provide early results, which can then be used to inform further analysis, resulting in reduced investigative times. Roussev and Quates [12] show that one way to achieve this is to use more sophisticated processing techniques, with extensive use of parallel processing facilitating quick initial assessments. A case study is presented which evaluates the effectiveness of *sdhash* in correlating evidence across a variety of devices. Full disk hashes and bytewise similarity hashes are calculated in parallel, which alleviates the initial wait time required to forensically copy a disk. This method avoids the file system by accessing data directly on the disk, and proved to be effective with a multi-processor workstation.

Roussev et al. [13] discuss a parallel processing model intended to allow data to be processed as fast as it can be acquired from the storage media. Individual analyses, such as hashing, parsing windows registry entries, text indexing, and file decompression, are handled by separate worker nodes, allowing for distributed analysis. However, the authors noted that a typical 8-core workstation lacked the computational power required to process data in real time, except when calculating traditional cryptographic hashes or parsing small quantities of data. Drastically more computational power, or data reduction techniques, are required for a complete real time analysis of a disk stream.

Garfinkel [14] describe the *bulk_extractor* tool, which uses a parallel processing model, to extract forensically relevant information from the disk. Scanners were developed which extract images, documents, and textual data such as email addresses, telephone numbers, and credit card numbers. Information critical to the investigation can then be identified via an overview which provides histograms of potentially important data, such as repeated uses of a particular email address.

Rather than focusing on processing large volumes of data in a timely fashion, Penrose et al. [15] reduce the quantities of data to process by sampling small blocks from the storage media, while maximising sequential throughput from the device. Traditional cryptographic hashing is applied at the block level, with bloom filter look ups being used to detect fragments identified from known contraband. The choice of block size is critical, as block alignment and non-probative blocks are a concern, as in Garfinkel et al. [4]. However, this method proved to be effective for contraband detection with high statistical confidence, allowing for SSDs to be sampled in seconds, and hard disk drives in under an hour in most cases.

Triage can be effected by increasing the utilisation of computational resources, however this is fundamentally limited by the read speed of the storage media. Probabilistic sampling works within the same constraints, but acknowledges that it is not necessary to read all data on a device to detect contraband.

III. Approach

In contrast to prior work in digital forensic triage, the approach described in this work is not probabilistic, nor does it require extensive computational power. Instead it is acknowledged that not all data on the storage media needs to be processed on a device and applies this concept at the file level. We explore different strategies for using traditional cryptographic hashing algorithms, such as SHA256, to extract signatures from partial files in an efficient manner. Unlike prior work in sub-file signature generation [10], [11], this approach is file type agnostic. However, as image processing is an integral part of many digital forensics investigations, the experimental datasets chosen are comprised of images.

A. Sub-file Hashing Strategies

Three distinct sub-file approaches were used in this work, with the generalised form containing a parameter, n, indicating the number of bytes constituting a read block. This data is then hashed using the SHA256 algorithm to produce a file signature.

[First n]: Read n bytes from the **beginning** of the file. This was exemplified by 4KiB and 80KiB in this work. 4KiB was chosen as it represents the fastest possible sub-file hash, corresponding to the smallest read unit on contemporary storage drives for both solid state and mechanical media. It is also a good approximation for the performance of prior sub-file signature generation schemes [10], [11]. The higher boundary of 80KiB (20×4KiB data blocks) corresponds to approximately $2/3$ of the mean file size of images in the Flickr 1 Million dataset, and is chosen to represent the largest chunk of a file which could still be considered to be 'sub-file' for most images. These read blocks are always expected to be byte aligned with modern hard disk sectors and SSD pages, which are 4096 bytes.

[Last n]: Read n bytes from the **end** of the file, exemplified by 4KiB and 12KiB in this work. The former value is chosen to maximise speed, while the latter was chosen to highlight the performance trade-off when the size is increased slightly by one or more disk blocks. These read blocks are not expected to align with disk sectors or SSD pages often ($1/4096$ of the time), and in practice two or four sectors/pages, respectively, would be accessed by the underlying storage media.

[First n+Last n]: Read n bytes from **both** the start and the end of the file, resulting in twice the amount of data to be read as the previous two methods. This technique is used both as a method to potentially improve discriminative power, but also to highlight block retrieval performance characteristics of non-contiguous data blocks. n is 4KiB in this work, and was chosen to represent the most effective performance/discrimination trade-off. This results in 8KiB of data to hash, and often three disk sectors/SSD pages to be accessed by the storage media. This strategy is abbreviated to First+Last n in the following discussion.

B. Datasets

Two datasets were used in this work: *i)* The Flickr 1 Million dataset [5], comprised of 1 million JPEG images retrieved from the Flickr image hosting platform, and *ii)* The Govdocs dataset [6], originally comprised of approximately 108,000 JPEG images, converted to the PNG format. Images were converted to PNG using the Python Pillow library [16], discarding four images which did not convert correctly, and removing duplicates as determined by the SHA256 algorithm.

Dataset	Size Range	Median	Mean	Total
Flickr 1 Million	8KiB – 1.5MiB	117KiB	124KiB	120GiB
Govdocs PNG	170B – 38.2MiB	344KiB	1.4MiB	148GiB

TABLE I
FILE SIZE STATISTICS FOR THE FLICKR 1 MILLION AND GOVDOCS PNG DATASETS USED IN THIS WORK. BOTH DATASETS HAVE SIMILAR TOTAL SIZE, HOWEVER THE GOVDOCS DATASET HAS APPROXIMATELY $1/10th$ AS MANY IMAGES.

This created a dataset with much larger file sizes than the original, allowing for contrasts to be made with the Flickr dataset, which contains much smaller files. File size statistics for both datasets are provided in Table I.

C. Benchmark Set-up

In order to determine the potential processing speed improvements of each sub-file hashing strategy, a direct comparison is made with traditional full file hashing using the SHA256 algorithm, referred to in this work as Fullhash. Timed benchmarks were carried out on a workstation (i5-4690k, 16GiB DDR3 RAM) running Ubuntu 16.04 LTS, across both a hard disk (Western Digital Red 4TB) and SSD (Crucial MX300 525GB). Neither drive hosted the Operating System, and the SSD was tested using the NTFS and EXT4 file systems, as performance differences were noted in the literature [11]. Code was written in Python, and executed in the Python 2.7.12 interpreter. The built-in `hashlib` library, which performs SHA256 calculations, is implemented in C on the back-end. Datasets were copied to the drives sequentially, and file orders were determined by Python's `os.listdir` function. Volumes were mounted with the `-ro,noatime` flags to prevent modification. In copying files to an empty drive sequentially, and using their ordering provided by the file system, sequential read performance is maximised. That is, the experiments here provide the best case scenarios for sequential read, and therefore full file hashing, constituting a strong baseline to compare sub-file techniques against. Reported times do not include file enumeration times from `os.listdir`.

IV. Findings

There are two important evaluation criteria for sub-file hashing strategies: i) they must be highly discriminative, allowing for contraband to be detected with a high degree of accuracy, and *ii)* they must be considerably faster than traditional full file hashing in order to justify the trade-off in discriminating power. Both of these criteria are discussed in turn.

A. Sub-file Signature Discriminating Power

The very nature of a signature demands that it be unique, or almost unique, in the context of its use. In digital forensics a higher degree of discriminating power results in a lower number of false positives. Table II shows the number of unique signatures for each sub-file hashing strategy on each dataset, not including full file duplicates.

Number of Unique Signatures				
Dataset	**First 4KiB**	**First 80KiB**	**First+Last 4KiB**	**Last 4KiB**
Flickr 1 Million (no Fullhash duplicates)	970633 (97.10%)	999349 (99.97%)	999622 (100%)	999622 (100%)
Govdocs PNG	108670 (99.80%)	108824 (99.94%)	108885 (100%)	108885 (100%)

TABLE II
THE NUMBER OF UNIQUE SIGNATURES FOR VARIOUS SUB-FILE HASHING STRATEGIES ACROSS BOTH DATASETS. FIRST 80KIB INCLUDED TO SHOW THAT EVEN READING A SUBSTANTIAL PART OF THE BEGINNING OF THE FILE DOES NOT PRODUCE UNIQUE VALUES.

The results show that hashing from the start of the file is highly discriminative with as little as 4KiB, however many bytes are required before the signatures would become unique, with 80KiB of data failing to produce completely unique signatures for both datasets. Hashing 4KiB from the end of the file, however, proved to be unique across both datasets, precluding the need to process larger data blocks from the end of the file. The first and last strategy is also unique as it contains the Last 4KiB, with the possibility that it may scale better to tens, or hundreds, of millions of files.

Without considering the performance trade-offs, reading from the end of the file, or both the start and the end, are the preferred options for high discriminative power. First+Last 4KiB is retained for performance analysis as it is potentially a more robust mechanism for increasing discriminative power than reading more blocks from the end of a file. Additionally, it provides an insight into the performance characteristics of acquiring non-contiguous small data blocks on a device.

B. Sub-file Signature Performance

Sub-file hashing strategies must be appreciably faster than full file hashing to be useful, and should be effective across different media types and file sizes. All hashing strategies were applied to both datasets for varying thread counts, with each combination being repeated 3 times, clearing memory caches between runs. Performance metrics are provided in Table III for the Flickr 1 Million dataset, and Table IV for Govdocs PNG. The left-hand column for each technique indicates the mean total time (seconds) for processing the dataset, while the right-hand side contains performance factors, obtained by dividing the time taken for sub-file strategies by the full file hashing times. A performance factor of 10 means that the approach is 10× faster than full file hashing. The Fullhash technique has no performance factor, as it would simply be compared to itself. The First 80KiB strategy was omitted as it has little to gain over reading the First 4KiB of the file, while 12KiB of data from the end of the file was tested to determine the impact of additional small data blocks on the most promising technique. Hard disk data is provided only for NTFS runs with a single thread, as multiple threads decrease performance in some cases, and sequential access is critical to maximal throughput on hard disks.

Several factors affect the performance of all approaches:

Drive Type: The largest factor affecting performance is the type of storage media, with sub-file approaches scaling much better on the SSD than the HDD. This is because the read head on a hard drive has to physically move to the appropriate track, and then wait for the disk platter to spin to the appropriate sectors before data can be retrieved. SSDs have no mechanical components, no seek times, and can fetch unrelated pages in parallel, resulting in much better small block throughput. This behaviour is captured by the random 4KiB read performance of the device, which is typically much lower than that of sequential read speeds. On the SSD random 4KiB read performance ranges from 6.6% to 50% of the sequential read speed, while the HDD lies between 0.4% and 1.5%, depending on queue depth and thread count, as measured by CrystalDiskMark 5.5 [17].

File Size: The size of the files being processed influences the behaviour a great deal. It appears that the overhead of non-sequential access on the HDD mean that sub-file strategies are actually less performant than simply reading the full file for small files, as with Flickr 1 Million. This file system overhead may be alleviated, as discussed below. However, when the file size is increased with the Govdocs PNG dataset, gains of 2–3× are possible, as this overhead is overwhelmed by the reduction in data. On the SSD, performance is reasonable even with small files, while increasing the mean file size by an order of magnitude results in similar performance gains, with all techniques performing well on the PNG dataset. The fraction of data, and therefore the data reduction ratio, of reading a single 4KiB block from each file depends on the file size. For Flickr 1 Million, this corresponds to 3% of the total data, while for Govdocs PNG it falls to 0.26%, resulting in a much higher tolerance for low 4KiB read throughput.

File System: EXT4 benchmark times were lower than NTFS across the board, with a particularly large performance difference for all sub-file strategies. This appears to indicate that the NTFS file system introduces substantial overheads when accessing small parts of a file, though the worst case SSD sub-file performance is still favourable when compared to sequentially accessing all file data. NTFS overheads can be avoided by parsing the MFT and obtaining a list of LBA addresses for files, which can then be used to sequentially order reads, resulting in greater performance [11]. This would likely bring the sub-file NTFS HDD performance in line with full file hashing for the Flickr Dataset.

No. Threads: All approaches scale well with thread count on the SSD, however NTFS appears to plateau around 8 threads, with no additional benefit beyond this. EXT4 continues to scale up to 32 threads, but returns appear to diminish somewhat after 16 threads. It should be noted that the workstation's processor has four cores and no hyper-threading.

The fastest technique, First 4KiB, achieves very good performance compared to full file hashing, and leads the timed benchmarks on all but the Flickr HDD run. However, it is followed closely by the Last 4KiB strategy, which achieved better discrimination in Section IV-A. Reading two further 4096 byte

Flickr 1 Million Benchmarks

Mean hash Time (s) and Relative Performance Factor to Fullhash

FS / Drive	**Threads**	**First 4KiB**		**First+Last 4KiB**		**Last 4KiB**		**Last 12KiB**		**Fullhash**
	1	307.3	2.6	703.3	1.2	481.3	1.7	571.8	1.4	813.0
	2	287.1	2.4	619.5	1.1	408.5	1.7	491.4	1.4	678.3
NTFS	4	275.7	2.5	542.6	1.3	347.5	2.0	448.7	1.5	690.8
SSD	8	270.2	2.7	488.7	1.5	305.5	2.4	405.7	1.8	724.1
	16	270.4	2.7	493.2	1.5	309.2	2.4	410.7	1.8	735.0
	32	271.6	2.8	524.9	1.5	319.7	2.4	411.6	1.9	766.7
	1	257.8	3.4	502.7	1.8	282.0	3.1	393.4	2.2	882.8
	2	139.3	4.1	282.0	2.0	153.2	3.7	225.6	2.5	565.0
EXT4	4	82.7	4.5	169.2	2.2	90.8	4.1	137.6	2.7	370.3
SSD	8	55.6	4.9	114.3	2.4	59.0	4.6	88.9	3.1	273.9
	16	45.6	5.8	89.5	2.9	42.7	6.1	63.3	4.1	262.1
	32	44.3	5.9	81.2	3.2	37.9	6.8	59.8	4.3	259.1
NTFS HDD	1	1593.2	0.8	1587.0	0.8	1605.6	0.8	1621.3	0.8	1284.9

TABLE III

Benchmark results to read and hash (SHA256) for the Flickr 1 Million Dataset. Mean time in seconds on the left hand side for each technique, followed by the colour coded performance factor on the right. Performance factors are relative to Fullhash times, with a factor of 10 indicating the sub-file technique is ten times faster. These were calculated by dividing the time taken for full file hashing by the time taken for the sub-file technique.

Govdocs PNG Benchmarks

Mean hash Time (s) and Relative Performance Factor to Fullhash

FS / Drive	**Threads**	**First 4KiB**		**First+Last 4KiB**		**Last 4KiB**		**Last 12KiB**		**Fullhash**
	1	30.7	25.1	94.0	8.2	48.5	15.9	55.8	13.8	770.3
	2	27.9	21.4	74.4	8.0	39.1	15.2	47.6	12.5	595.3
NTFS	4	27.4	22.3	57.6	10.6	36.6	16.7	43.6	14.0	609.8
SSD	8	26.9	22.1	49.2	12.1	31.4	18.9	40.0	14.8	593.4
	16	26.8	22.1	50.0	11.9	32.5	18.3	40.2	14.8	593.8
	32	27.7	22.0	53.0	11.5	33.6	18.1	40.8	14.9	609.0
	1	28.8	25.7	53.6	13.8	31.0	23.8	43.0	17.2	739.3
	2	15.3	32.7	30.0	16.6	17.0	29.4	24.4	20.5	500.1
EXT4	4	9.0	39.7	18.0	19.8	10.0	35.7	14.9	24.0	357.8
SSD	8	6.0	54.2	12.0	26.8	6.5	49.9	9.5	33.9	323.1
	16	4.8	65.9	9.2	34.3	4.6	68.6	6.6	47.6	315.2
	32	4.7	66.6	8.0	38.7	4.4	70.6	5.8	53.4	310.5
NTFS HDD	1	691.6	2.6	818.9	2.2	665.9	2.7	673.6	2.6	1776.8

TABLE IV

Benchmark results to read and hash (SHA256) for the Govdocs PNG Dataset. Mean time in seconds on the left hand side for each technique, followed by the colour coded performance factor on the right. Performance factors are relative to Fullhash times, with a factor of 10 indicating the sub-file technique is ten times faster. These were calculated by dividing the time taken for full file hashing by the time taken for the sub-file technique.

File Size	**Fullhash Time**	**Last 4KiB Improvement Factor**
(Flickr) 123 KiB	2.5 ms	7.9×
400 KiB	5.4 ms	17.0×
800 KiB	9.6 ms	30.1×
1 MiB	12.0 ms	37.4×
(Gov. PNG) 1.4 MiB	16.2 ms	50.8×
10 MiB	108.4 ms	339.4×

TABLE V

Linear Regression predictions for the time taken to read and hash full files of various sizes, with the relative benefit of Last 4KiB hashing. Figures are derived from the same data as Figure 1 using the SSD with EXT4. Improvement factor is calculated by dividing the Fullhash time by the constant time to acquire and hash Last 4KiB (0.32ms).

chunks from the end of the file results in some performance degradation, but nothing substantial. The worst performing sub-file hashing strategy, First+Last 4KiB performs an order of magnitude faster than full file hashing on the SSD for the PNG dataset, but barely edges it out on the Flickr dataset.

All sub-file approaches look viable on relatively large files, but based on these findings it appears that the Last 4KiB approach would be the best choice in terms of performance and discrimination. On the SSD, this method approaches a factor of 20× over full file hashing on NTFS, and an enormous 70× on EXT4. The Last 4KiB approach performs similarly to prior file type specific sub-file hashing schemes [10], [11], while generalising the approach to any file type.

C. Performance Scaling With File Size

As file size has such a large influence on the performance of sub-file hashing strategies, it was explored in more detail by acquiring per-file benchmark estimates. In order to control for the potential variance in measuring small IO operations, files were grouped into bins of width 8192 bytes, discarding bins with fewer than eight files. The average times for full file hashing and Last 4KiB were then calculated from each bin on the SSD with the EXT4 file system. Figure 1 is a scatter plot of average per-file times for each bin.

The time taken to acquire the Last 4KiB of each file is essentially a fixed cost which does not scale with file size, with a mean of 0.32ms in this experiment. Full file hashing, on the other hand, unsurprisingly scales linearly with the size of the file, easily taking tens of milliseconds to process when file sizes are in the Mebibytes.

Linear regression was then used to generate a predictive model to obtain hash times and their respective Last 4KiB performance factors, depicted in Table V. The equation for these predictions, where x is the byte size of the file, is as follows:

$$Fullhash\_time(s) = 1.02196 \times 10^{-8}x + 1.23139 \times 10^{-3}$$

These predictions match up fairly well with the SSD EXT4 benchmarks with 32 threads, and therefore serve as a reasonable indicator of performance. Files with 10MiB of data, which may include very high resolution JPEGs, uncompressed images, or short videos, would enjoy a performance increase factor over 300×. This level of scaling means that when file sizes are in the Mebibytes, sub-file hashing strategies should perform much better than full file hashing, regardless of storage device or file system.

D. Impact of Full File Confirmation Hashing

The Last 4KiB strategy generated unique signatures for all unique files across both datasets, however, some conditions may change this, with homogeneous data, similar files, or very large volumes of images. This can be tackled by reading more data from the end of the file, as with the Last 12KiB approach, or using the less performant First+Last 4KiB strategy. In some use cases it may still be prudent to fully hash the entire file to confirm the original detection assessment. In order to assess the impact of confirmation hashing, worst case scenario estimates are provided in Table VI for 1% and 10% confirmation rates. These estimates ignore disk or memory caching for the file, which in reality may reduce the confirmation overhead.

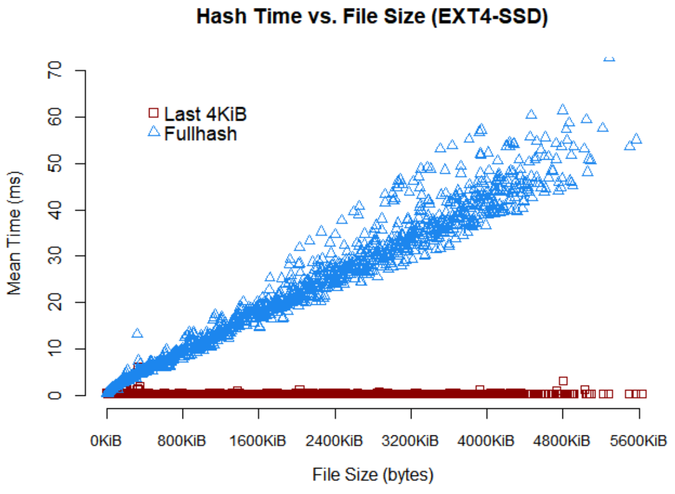

Fig. 1. **File size plotted against hash time for Last 4KiB and full file hashing techniques. Mean file times are shown for file size bins of 8192 bytes, carried out single threaded on the SSD with EXT4.**

Even at a 10% confirmation rate, where 10% of files are fully hashed, the only scenario which is slower than full file hashing is on the hard disk with Flickr 1 Million. The best case for Govdocs PNG and EXT4 is still an order of magnitude faster than fully hashing all files, with the worst case for Flickr on NTFS still taking around half of the time.

It is important to note that confirmation hashing does not need to be conducted at the same time as the initial detection scan. Sub-file strategies may be deployed to quickly process the data, with confirmation hashing occurring after other up-front processing has been conducted. In this sense, initial results can be obtained quickly, with confirmation carried out in the background afterwards. This fits in well with the triage models in the literature, where early results are critical to decision making.

V. Conclusions and Future Work

This work has explored the potential of sub-file forensic hashing strategies, where small fragments of a file are read from the storage media and cryptographically hashed, as a means of quickly detecting contraband files. The results show that sub-file hashing is highly discriminative, with reading 4KiB of data from the end of the file producing a unique signature for 1.1 million images in this experiment. Performance benchmarks show that various sub-file hashing strategies are much faster than full file hashing on solid state drives, with increases of up to 70×. Hard disk performance relative to reading the entire file varies greatly with file size, as random 4KiB read performance is much lower on these devices. The choice of file system is also shown to have a large impact, with the approach working better on EXT4 than NTFS. Sub-file hashing strategies are effective, and scale very well with

Impact of Full file confirmation hashing on Last 4KiB lookups
Time in seconds

Dataset	**FS / Drive**	**Threads**	**Last 4KiB Base**	**Last 4KiB 1% Hit Rate**	**Last 4KiB 10% Hit Rate**	**Fullhash**
Flickr 1 Million	NTFS SSD	32	319.7	327.4	396.4	766.7
	EXT4 SSD	32	37.9	40.5	63.8	259.1
	NTFS HDD	1	1605.6	1618.4	1734.1	1284.9
Govdocs PNG	NTFS SSD	32	33.6	39.7	94.5	609
	EXT4 SSD	32	4.4	7.5	35.45	310.5
	NTFS HDD	1	665.9	683.7	843.6	1776.8

TABLE VI
THE WORST CASE SCENARIO TIMES FOR CONFIRMING A LAST 4KIB LOOKUP WITH FULL FILE HASHING, FOR 1% AND 10% DETECTION RATES. CALCULATED BY ADDING PERCENTAGE OF FULLHASH ON TO LAST 4KIB TIMES.

file size, such that the time taken to process high resolution images, or small video files, may be two orders of magnitude faster on an SSD.

Many consumer devices rely solely on solid state technology for their internal storage, particularly in the mobile and laptop space. This work lays the foundation for digital forensic techniques which take advantage of the performance characteristics of NAND devices, greatly reducing the time taken to conduct investigations and alleviating law enforcement backlogs.

Future work in this area should explore the potential for this approach on external media, mobile phones, and networked storage. An additional opportunity may be found in conducting experiments on recent storage developments, such as the NVMe protocol or Intel Optane devices, which have much higher performance than prior NAND media using SATA interfaces. Hard disk drives are likely to remain in use cases where high storage capacities are critical, however the forensics community should prepare for the eventuality that magnetic media may be a rarity in a typical investigation, and adjust our processing approaches accordingly.

ACKNOWLEDGEMENT

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

REFERENCES

[1] D. Quick and K.-K. R. Choo, "Impacts of increasing volume of digital forensic data: A survey and future research challenges," *Digital Investigation*, vol. 11, no. 4, pp. 273–294, Dec. 2014. [Online]. Available: http://linkinghub.elsevier.com/retrieve/pii/S1742287614001066

[2] D. Lillis, B. Becker, T. O'Sullivan, and M. Scanlon, "Current Challenges and Future Research Areas for Digital Forensic Investigation," *arXiv:1604.03850 [cs]*, Apr. 2016, arXiv: 1604.03850. [Online]. Available: http://arxiv.org/abs/1604.03850

[3] N. Beebe, "Digital forensic research: The good, the bad and the unaddressed," in *IFIP International Conference on Digital Forensics*. Springer, 2009, pp. 17–36. [Online]. Available: http://link.springer.com/chapter/10.1007/978-3-642-04155-6_2

[4] S. Garfinkel, A. Nelson, D. White, and V. Roussev, "Using purpose-built functions and block hashes to enable small block and sub-file forensics," *Digital Investigation*, vol. 7, pp. S13–S23, Aug. 2010. [Online]. Available: http://linkinghub.elsevier.com/retrieve/pii/S1742287610000307

[5] M. J. Huiskes, B. Thomee, and M. S. Lew, "New trends and ideas in visual concept detection: the MIR flickr retrieval evaluation initiative," in *Proceedings of the international conference on Multimedia information retrieval*. ACM, 2010, pp. 527–536. [Online]. Available: http://dl.acm.org/citation.cfm?id=1743475

[6] S. Garfinkel, P. Farrell, V. Roussev, and G. Dinolt, "Bringing science to digital forensics with standardized forensic corpora," *Digital Investigation*, vol. 6, pp. S2–S11, Sep. 2009. [Online]. Available: http://linkinghub.elsevier.com/retrieve/pii/S1742287609000346

[7] J. Kornblum, "Identifying almost identical files using context triggered piecewise hashing," *Digital Investigation*, vol. 3, Supplement, pp. 91–97, Sep. 2006. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1742287606000764

[8] V. Roussev, "Data fingerprinting with similarity digests," in *IFIP International Conference on Digital Forensics*. Springer, 2010, pp. 207–226.

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

[10] S. McKeown, G. Russell, and P. Leimich, "Fast Filtering of Known PNG Files Using Early File Features," in *Annual ADFSL Conference on Digital Forensics, Security and Law*, Daytona Beach, Florida, USA, 2017. [Online]. Available: https://commons.erau.edu/adfsl/2017/papers/1/

[11] S. McKeown, G. Russell, and P. Leimich, "Fingerprinting JPEGs With Optimised Huffman Tables," *Journal of Digital Forensics, Security and Law*, In Press.

[12] V. Roussev and C. Quates, "Content triage with similarity digests: The M57 case study," *Digital Investigation*, vol. 9, pp. S60–S68, Aug. 2012. [Online]. Available: http://linkinghub.elsevier.com/retrieve/pii/S1742287612000370

[13] V. Roussev, C. Quates, and R. Martell, "Real-time digital forensics and triage," *Digital Investigation*, vol. 10, no. 2, pp. 158–167, Sep. 2013. [Online]. Available: http://linkinghub.elsevier.com/retrieve/pii/S1742287613000091

[14] S. L. Garfinkel, "Digital media triage with bulk data analysis and bulk_extractor," *Computers & Security*, vol. 32, pp. 56–72, Feb. 2013. [Online]. Available: http://linkinghub.elsevier.com/retrieve/pii/S0167404812001472

[15] P. Penrose, W. J. Buchanan, and R. Macfarlane, "Fast contraband detection in large capacity disk drives," *Digital Investigation*, vol. 12, Supplement 1, pp. S22–S29, Mar. 2015. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1742287615000080

[16] A. Clark, "Python-Pillow," 2015. [Online]. Available: https://github.com/python-pillow/Pillow

[17] N. Miyazaki, "CrystalDiskMark," 2017. [Online]. Available: https://crystalmark.info/software/CrystalDiskMark/manual-en/