

May 16th, 11:00 AM

## Understanding Deleted File Decay on Removable Media using Differential Analysis


James H. Jones Jr  
*George Mason University, jjonesu@gmu.edu*

Anurag Srivastava  
*George Mason University, asrivas4@masonlive.gmu.edu*

Josh Mosier  
*George Mason University*

Connor Anderson  
*George Mason University*

Seth Buenafe  
*George Mason University*  
Follow this and additional works at: <https://commons.erau.edu/adfsl>

 Part of the [Computer Law Commons](#), [Data Storage Systems Commons](#), [Forensic Science and Technology Commons](#), [Information Security Commons](#), and the [OS and Networks Commons](#)

---

### Scholarly Commons Citation

Jones, James H. Jr; Srivastava, Anurag; Mosier, Josh; Anderson, Connor; and Buenafe, Seth, "Understanding Deleted File Decay on Removable Media using Differential Analysis" (2017). *Annual ADFSL Conference on Digital Forensics, Security and Law*. 13.  
<https://commons.erau.edu/adfsl/2017/papers/13>

This Peer Reviewed Paper is brought to you for free and open access by the Conferences at Scholarly Commons. It has been accepted for inclusion in Annual ADFSL Conference on Digital Forensics, Security and Law by an authorized administrator of Scholarly Commons. For more information, please contact [commons@erau.edu](mailto:commons@erau.edu).

**EMBRY-RIDDLE**  
Aeronautical University™  
SCHOLARLY COMMONS

(c)ADFSL



# UNDERSTANDING DELETED FILE DECAY ON REMOVABLE MEDIA USING DIFFERENTIAL ANALYSIS

James H. Jones, Jr.<sup>1</sup>, Anurag Srivastava<sup>2</sup>, Josh Mosier<sup>3</sup>, Connor Anderson<sup>3</sup>, Seth Buenafe<sup>3</sup>

<sup>1</sup>George Mason University  
Department of Electrical and Computer Engineering  
Fairfax, Virginia 22030  
jjonesu@gmu.edu

<sup>2</sup>George Mason University  
Volgenau School of Engineering  
Fairfax, Virginia 22030  
asrivas4@masonlive.gmu.edu

<sup>3</sup>George Mason University  
Aspiring Scientists Summer Internship Program  
Fairfax, Virginia 22030

## ABSTRACT

Digital content created by picture recording devices is often stored internally on the source device, on either embedded or removable media. Such storage media is typically limited in capacity and meant primarily for interim storage of the most recent image files, and these devices are frequently configured to delete older files as necessary to make room for new files. When investigations involve such devices and media, it is sometimes these older deleted files that would be of interest. It is an established fact that deleted file content may persist in part or in its entirety after deletion, and identifying the nature of file fragments on digital media has been an active research area for years. However, very little research has been conducted to understand how and why deleted file content persists (or decays) on different media and under different circumstances. The research reported here builds upon prior work establishing a methodology for the study of deleted file decay generally, and the application of that methodology to the decay of deleted files on traditional computing systems with spinning magnetic disks. In this current work, we study the decay of deleted image files on a digital camera with removable SD card storage, and we conduct preliminary experiments for direct SD card and USB storage. Our results indicate that deleted file decay is affected by the size of both the deleted and overwriting files, overwrite frequency, sector size, and cluster size. These results have implications for digital forensic investigators seeking to recover and interpret file fragments.

**Keywords:** digital forensics; digital trace; file fragment; residual content; deleted data persistence; deleted file decay

## 1. INTRODUCTION

The use and operation of digital devices creates associated digital traces: files and data on storage devices, data in volatile memory, network traffic, Windows registry keys, CPU register values, etc. These traces, in whole and in part, are used by digital forensics investigators to infer and reconstruct past events, and are also harvested by criminal actors to collect private information. The persistence and decay of digital traces over time varies based on the type of trace, the storage or processing medium, and other inadvertent or deliberate activity which may damage or destroy the trace. While some traces may remain intact over time, most deleted or otherwise released content is altered, destroyed, and disassociated over time due to normal system operation and deliberate obfuscation activity. The current state of practice in digital forensics accepts that traces and trace fragments may or may not be available. Analysts are able to reason over the traces that are available, and in specific cases may attempt to explain the cause and significance of trace presence or absence; however, the state of the art has not addressed, in a rigorous and generalizable manner, the question of why trace or trace fragments do or do not persist or decay.

Most digital traces, such as allocated files, configuration settings, running processes, or network traffic are designed to persist while they are allocated or in use; however, once de-allocated, no longer active, or passed to another process, these traces are subject to decay, modification, and destruction. For example, a file is stored on a magnetic hard disk or solid-state media in groups of sectors called clusters. A sector is the smallest unit that a block device can read or write in a single operation, and a cluster is the smallest unit that a file system can allocate or de-allocate. Clusters are often multi-sector and

aligned, but not necessarily. When a file is deleted, the file system marks the clusters storing the data as available for future use as needed, but does not typically alter any of the original file data in those clusters. The operating system and its file system may eventually overwrite some or all of the media sectors used to store the original file. As a result, some or all of the deleted file will be destroyed, while other parts may remain intact for an indefinite period of time. Tools exist to "undelete" files when the data remains intact, file carving techniques (Ravi et al, 2016) (Yi et al, 2015) (Garfinkel, 2007) can recover full or partial files after deletion, and forensic investigators regularly recover full and partial traces from media. While solid state hard disks (SSDs) and thumb drives may use the same file systems as magnetic hard disks, the underlying operation of these devices is significantly different. The flash memory of SSDs and thumb drives has an additional processing layer, called the Flash Translation Layer (FTL), which is designed to optimize the reliability, performance, and lifetime of the device. FTL implementations are generally not published and vary across vendors and device types, although recent work reverse engineers the firmware of flash implementations to reconstruct data (Zhang, 2015). The FTL and associated device logic implement wear leveling (writing to all locations an even number of times), TRIM (preemptively erasing storage locations so they are ready for a subsequent write), and proactively rearrange the data within the storage device. As a result, deleted file persistence on magnetic and flash devices is significantly different, although TRIM implementations are quite fragile and frequently do not work as designed (Gubanov & Afonin, 2014), leaving more data available for recovery than expected.

To help visualize deleted file decay and partial trace recovery, a deleted BMP image

file was repeatedly rendered as sectors of the original file were overwritten. This sequence of images is shown in Figure 1, where 100% of the sectors are intact on the left, and sectors are overwritten going from left to right until only 15% of the original sectors remain intact for the rightmost image. The BMP image format is most suitable for such recovery, as file contents map directly to the image layout, although other work (Sencar & Memon, 2009) (Uzun & Sencar, 2015) has shown that partial image recovery from other image formats is

possible as well. For arbitrary file types, work by Garfinkel and McCarrin (2015) investigates the probative potential of file fragments. The work presented here helps to understand the factors affecting deleted file decay on flash media, and is relevant to digital forensic investigations as practitioners extend their capabilities beyond whole file recovery and into partial file recovery and interpretation.

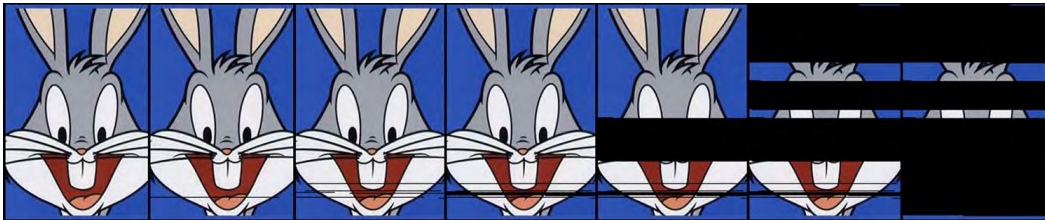


Figure 1. Rendering deleted BMP file as it decays

## 2. RELATED WORK

Fairbanks and Garfinkel (2012) posited factors which might affect the persistence of deleted file content. This paper is predated by other work observing the effects of data persistence but not attempting to explain it beyond the immediate case. In chapter 7 of their 2005 book "Forensic Discovery," Farmer and Venema (2005) published experimental data and partial explanations for the persistence of deleted file information. While useful, the experiments were limited in scope, and the discussion sought to explain the observed persistence given aspects of their particular test system rather than computer systems in general. As noted by Venema elsewhere (Reust & Friedburg, 2006), "...persistence of deleted file content is dependent on file system, activity, and amount of free space (a complex relationship)." Roussev and Quates (2012) tangentially show the effects of data persistence in a case study of the M57 dataset. The case study focused on content triage using similarity digests, but the paper includes a

graph of deleted file data persistence for a specific example (see Figure 1 on page S66 of that publication). The decay effect, observed as an almost linear reduction in deleted file content over a matter of days, is explained by the user deleting the files and continuing to use the system normally. Had the user in this case employed a well-implemented secure deletion tool, as discussed in (Joukov et al, 2006) and elsewhere, the original file data locations would have been overwritten and rendered irrecoverable immediately upon deletion. Such tools typically do not address the possibility of data remnants in locations other than the primary storage clusters, meaning that remnant recovery is possible even in the face of secure deletion. An example of this is when a new copy of a file is made during the modification process and the original version is then deleted. Although the new version will be securely erased, the persistence of the clusters that constituted the original version is unknown. This can also happen in the absence of file modification due

to drive defragmentation. The challenge in such cases, and in any experiment or investigation without ground truth, is how to establish that a remnant is part of a specific original file vs. a false positive, a situation which is discussed in (Garfinkel et al, 2010). The work proposed here will track the sectors of a deleted file in place, although integrating this with other work finding and reasoning over deleted file fragments found in any location is a logical next step and will lead to a more complete model of deleted file persistence.

Factors affecting deleted file persistence noted by Fairbanks and Garfinkel include device types, especially the difference between magnetic and flash storage. Bell and Boddington (2010) wrote one of the first complete analyses of this effect for solid state drives. Later work, including (Casey & Turnbull, 2011) (Huang et al, 2015), discussed the impact of flash memory for digital forensics in the context of mobile devices and the recovery of fragmented files. Fairbanks and Garfinkel also suggest that file type may affect deleted file persistence, suggesting email databases as an example. As early as 2007, (Stahlberg et al, 2007) and (Litchfield, 2007) discussed deleted data persistence in databases and how this differs from normal file deletion. More recently, Conrad et al (2009) discussed deleted data persistence in the context of forensic analysis of a Sony PlayStation Portable. The rapid rise in cloud infrastructure, especially multi-tenant clouds, has generated interest in deleted data persistence in such environments where multiple entities share a common underlying infrastructure. Govan (2013) presents a detailed analysis of data remanence for several applications, and concludes that the effects are conflicting: deleted data may be replicated in multiple locations and so will be more recoverable, while the frequent writing of data

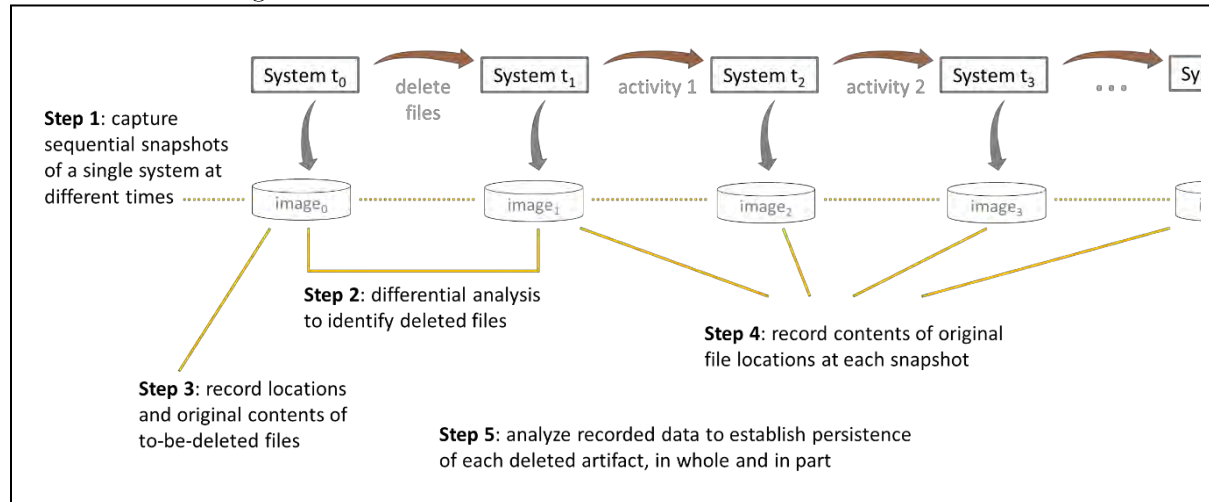
and lack of transparency will inevitably modify or destroy deleted file traces or at least call into question their integrity.

The work presented here is related to, but distinct from, efforts to identify the filetypes of recovered file fragments, work that has been ongoing for over 10 years (Li et al, 2006) (Calhoun & Coles, 2008) (Roussev & Quates, 2013). These efforts are aimed at identifying the type of file from which a fragment originated, whereas we are studying the factors affecting the decay of the original file into the fragments that are eventually recovered. Similarly, our work supports the development of deterministic approaches in digital forensics (Nagy et al, 2015), but is not itself a deterministic reasoning approach. While considerable effort has been spent to process full and partial digital traces, little work has been directed at understanding the mechanisms driving the decay of full traces into partial traces.

### 3. APPROACH AND METHODOLOGY

We developed and implemented a methodology to track the decay of deleted file contents over time. We first capture multiple sequential images of a device's stored data, where files to be tracked are deleted between the first two images (images 0 and 1). Files known to the file system in image 0 but not in image 1 are the deleted files. Once identified, we use image 0 to record the sector locations and original contents of those files (sectors) prior to deletion. We then track the contents of the original file's sectors over the remaining images, identifying if, when, and which sectors of each deleted file were changed. This approach is based on the differential analysis ideas articulated by Garfinkel et al (2012). The file and sector change data is then processed to form and test hypotheses as to which factors affect deleted file decay, and to design

additional experiments. This process is summarized in Figure 2 and detailed in the text that follows.



**Step 1:** Record sequential images of the raw hard disk contents for a single system over time. Files may be created prior to the first image at  $t_0$ , but the first image must include the files to be deleted and tracked as allocated files, and the files to track must be deleted between image 0 and image 1.

**Step 2:** Determine which files were deleted by comparing the allocated files in image 1 and image 0; files allocated in image 0 and not in image 1 are considered deleted. Deleted file information includes file name, file size, and the sectors allocated to the file in image 0.

**Step 3:** Record a cryptographic hash of the contents of the data sectors in image 0 (before deletion) for each deleted file.

**Step 4:** Record a hash of the contents of those same sectors for all other images (image 1 to image N).

**Step 5:** Use the stored data to analyze deleted file decay, i.e., when did the contents (sectors) of each file change and why.

We use a publicly available Python implementation of this methodology developed by Jones and Khan (2017). The

implementation assumes a series of disk images in raw format and a filesystem supported by The Sleuth Kit<sup>1</sup> (fiwalk). The implementation comes in two parts. The first program, `adiff.py`, processes the raw images and populates a sqlite3 database with sector hashes for each deleted file and image. The second program, `trace_file.py`, processes the sqlite3 database from `adiff.py` and produces one or more output items: data, graphs, and console displays representing deleted file decay.

For these experiments, we used FTK Imager and the `*nix dd` command to collect device and media raw images. Each device was prepared prior to any file activity according to the test design. Depending on the experiment, preparation included erasing old data, reformatting the device, wiping the data on the device, and/or configuring sector and cluster sizes. Files were then written to the media and a raw image of the media contents was recorded (image 0). Files were then deleted according to the specific experiment, where deletion may have been initiated by direct user action or by the system as a result of user

<sup>1</sup> <http://www.sleuthkit.org/sleuthkit/desc.php>

action (such as letting the media fill up and continue to write new files). Another raw image of the media contents was recorded after file deletion (image 1). Additional activity was then executed per the experiment design, and raw images of the media content were taken at designed intervals (images 2, 3, ...). The raw images were then processed using the implementation noted above to produce decay curves and raw decay data for each file. A sample decay curve for several files is shown in Figure 3, where the x-axis represents sequential media image identifiers, and the y-axis represents the % of the original file that remains intact at each image. Images are not necessarily taken at regular intervals, so the x-axis should be interpreted as representing experiment-specific activity and not equal time intervals. The % intact value is a fraction of the original file's sectors that remain unchanged. Raw output data consists of the per-file decay data as rows of comma-separated values, suitable for additional processing. The per-experiment sqlite database was also retained, allowing for additional queries and subsequent analysis based on characteristics such as file size, file type, file name and path, etc.

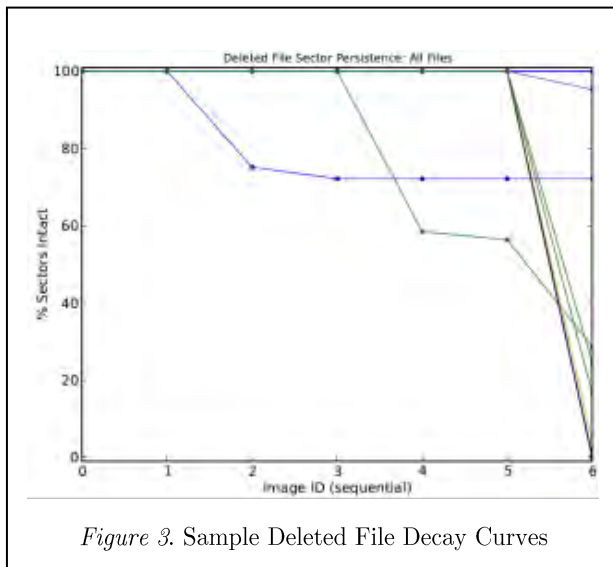


Figure 3. Sample Deleted File Decay Curves

## 4. RESULTS

We conducted two sets of experiments using the methodology described above. The first set of experiments used a security camera with SD card storage, and the second set of experiments used SD cards and USB sticks mounted on a Windows workstation.

### 4.1 Camera Experiment Results

A FosCam security camera was configured with 4 GB SD card storage, pointed out of a window, and pictures were taken automatically every few seconds until the media filled, at which point a raw image of the SD card storage was taken. The camera continued to take pictures, and by design began deleting and overwriting the original picture files. Images of the SD card media were taken at regular intervals as the device continued to take new pictures and the contents of the original picture files were repeatedly overwritten. A total of 4401 files were written over the original files. Figure 4 shows the new (overwriting) file sizes over time. Files are named and stored sequentially, so are ordered chronologically by name. Larger files were created as light and activity filled the camera's field of view in the morning, which occurred about 1/3 of the way into the graph.

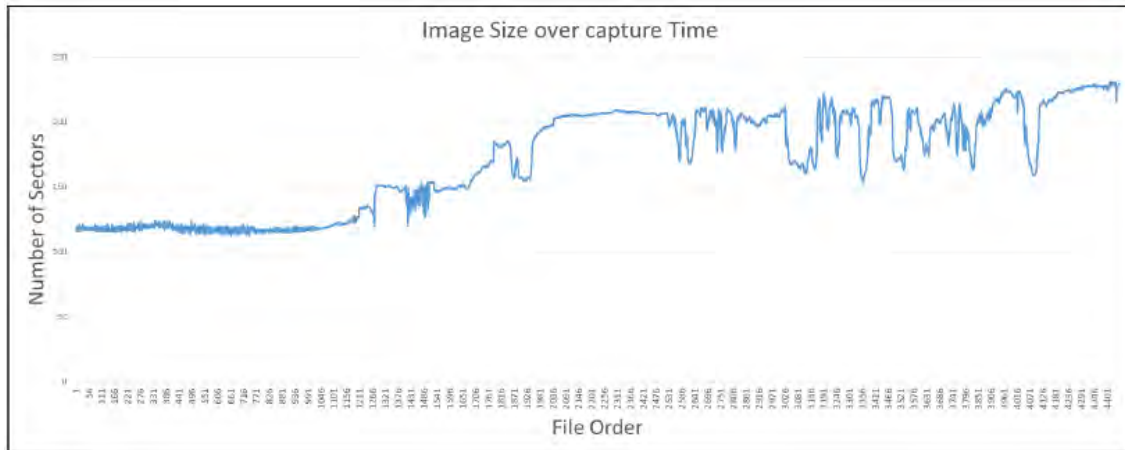


Figure 4. File size over time for overwriting camera files

Figure 5 shows individual deleted file persistence based on the order in which the file was originally written. These are the 157 files that were overwritten by the 4401 files in Figure 4. Note that the scales of Figure 4 and Figure 5 are not aligned; the files that were overwritten and tracked were stored sequentially on the media, and the overwriting files were stored sequentially as well, but we overwrote the original files multiple times. It is true that File 1 in both figures started at the same offset on the media, but varying file sizes means they don't necessarily end at the same offset. Consequently, File 2 in both graphs does not necessarily start or end at the same offset, etc. The percent persistence value (y-axis of Figure 5) is the final persistence of each file after multiple overwrites. The oscillating pattern (periodically ranging from  $\sim 0\%$  to  $\sim 50\%$ ) over sequential images may be explained by variation in the size of the overwriting files and the cluster size storing the deleted files. We used a cluster size of 32 kB (64 512-byte sectors), and the overwriting files oscillated around a cluster boundary (128 sectors, or two clusters). Overwriting files slightly smaller than two clusters would overwrite almost all of the data in the second and final cluster, whereas overwriting files slightly larger than two clusters would

overwrite very little of the data in the third and final cluster.



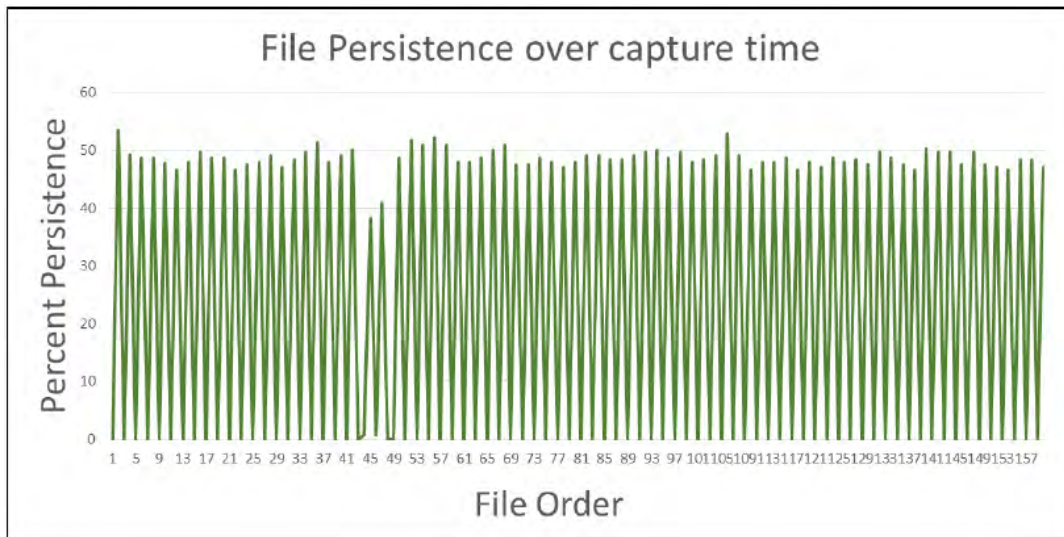


Figure 5. Persistence over time for deleted camera files

Figure 6 shows the distribution of deleted file persistence after one round of overwriting. Each line in the graph represents the persistence of one file, prior to deletion on the left-side y-axis and after the first overwrite on the right-side y-axis. At this early point, after one overwrite, the persistence of the deleted files ranges from ~0-50% with some files persisting at 100% (the values on the right-side y-axis). As the overwriting continued using different file sizes and multiple overwrites (about 25 in total), the distribution of deleted file persistence coalesced around 0% or 50%, with none of the values in between (as indicated in Figure 5). It appears that the relationship between original (deleted) file sizes, overwriting file sizes, and cluster sizes may explain this phenomenon, although we have yet to fully explore this.

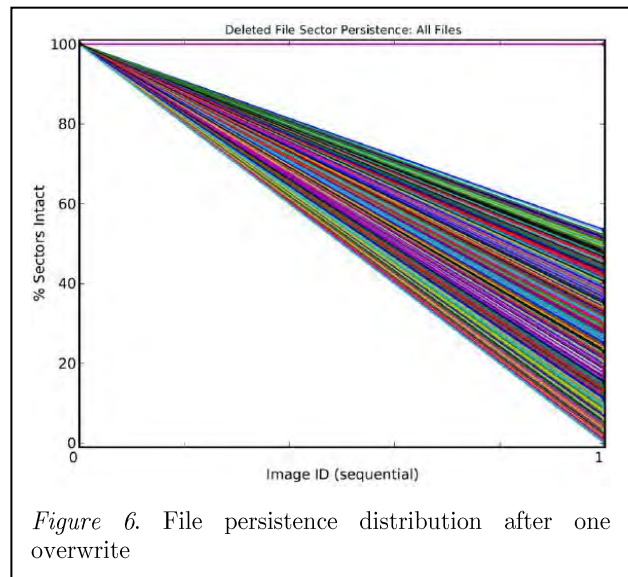


Figure 6. File persistence distribution after one overwrite

Figure 7 shows the relationship between file size and persistence for these same deleted files, also after only one overwrite. In the graph, data points are color-coded by clusters used: 2 cluster files are orange and use less than 128 sectors (x-axis), 3 cluster files are green and red and use from 128 to 192 sectors, and 4 cluster files are blue and use 192 to 256 sectors. The 3 cluster range (the x-axis from 128 sectors or 2 clusters to 192 sectors or 3 clusters) is further broken out to highlight a linear persistence pattern for some of these files

(the upward-sloping trend of  $> 0\%$  persistence between 140 sectors and 192 sectors; red points if viewing in color). Several interesting patterns are apparent (see the circled annotations on Figure 7): (i) the linear trend just noted from 140 sectors to 192 sectors, followed by a declining curve above 192 sectors, (ii) the narrowing shape from 110 sectors to 128 sectors, leading to the

concentrated flatline from 128 sectors to 155 sectors, (iii) the pattern of 0% and 100% which changes as the sector value changes, and (iv) the two non-zero clusters for files over 192 sectors. We are continuing to analyze this data and these patterns, and are continuing additional experiments, in order to explain these observations.

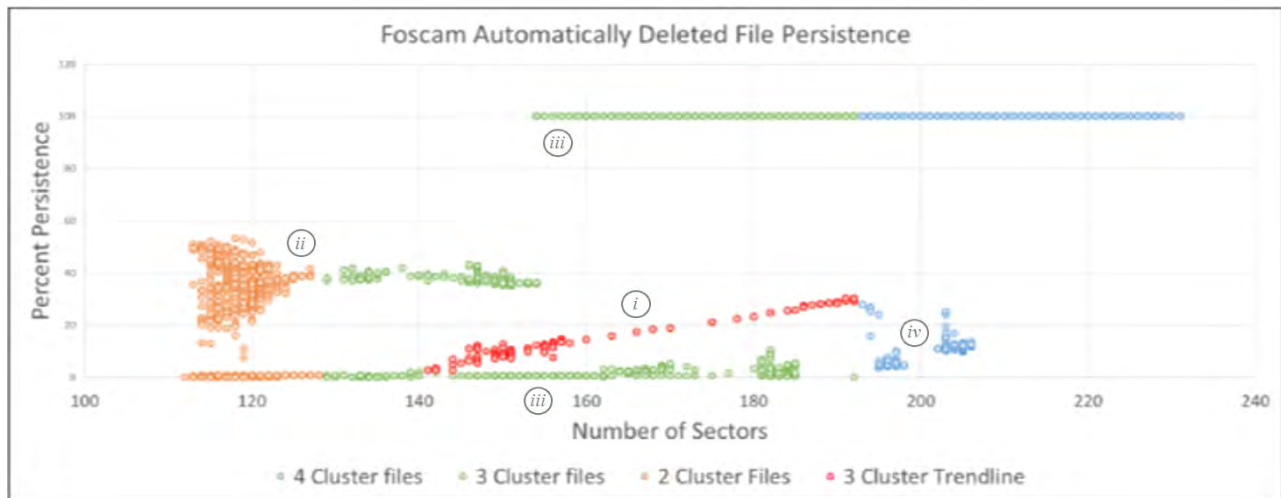


Figure 7: Final persistence vs. file size (number of sectors) for camera overwriting experiments

#### 4.2 SD Card and USB Experiment Results

A separate set of experiments was conducted to determine the effect of cluster size on deleted file persistence when multiple overwrites are performed. A common set of files of varying sizes was written to an SD card, once with 32-kB cluster size and once with 64-kB cluster size. For each configuration, the media was imaged (image 0), the files were deleted (image 1), and new files were repeatedly written to the media until full (images 2-6). Decay curves for the deleted files are shown in Figure 8. With a 64-kB cluster size, more deleted files have higher persistence after multiple overwrites. By comparison, a 32-kB cluster size yielded some deleted files of

very high persistence after 1-3 overwrites, but these files did not persist to the same degree as the 64-kB cluster size after subsequent overwrites. Average persistence across all files for each cluster size was modeled as exponential decay depending on time running, file size, media size, and new image rate (Figure 9).

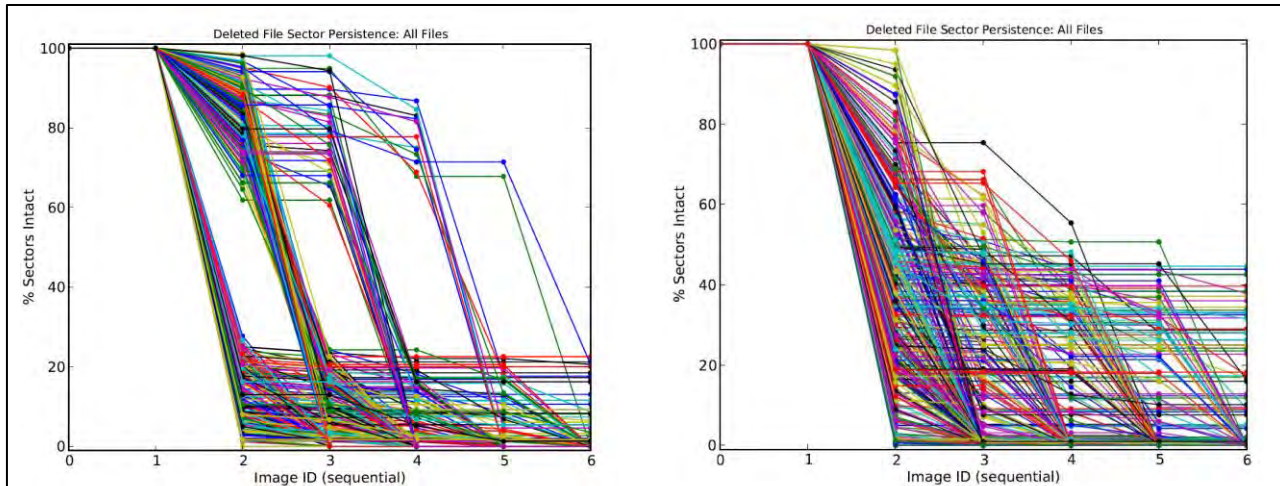


Figure 8. SD card file decay (cluster size 32 kB left and 64 kB right)

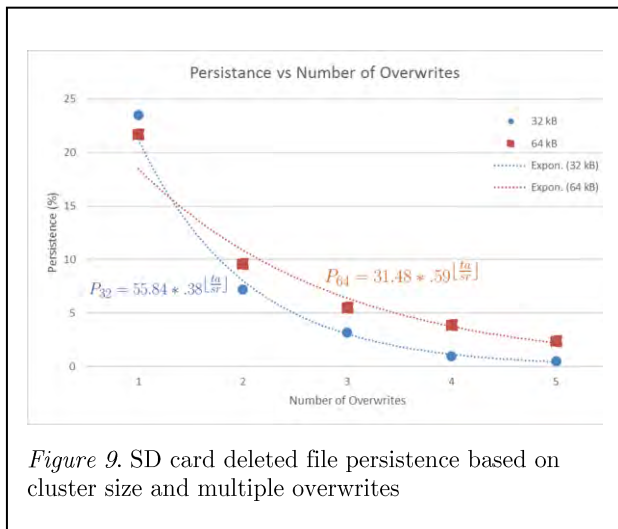


Figure 9. SD card deleted file persistence based on cluster size and multiple overwrites

Finally, preliminary experiments were run to explore the combined effect of file system, media, and cluster size on deleted file persistence. No effect was observed for different file systems (FAT32 vs. NTFS), but an effect was observed when the media type was USB and cluster size was varied (512 bytes, 4096 bytes, and 8192 bytes). See Figure 10 for associated decay curves. Additional experiments to explore this effect are underway.

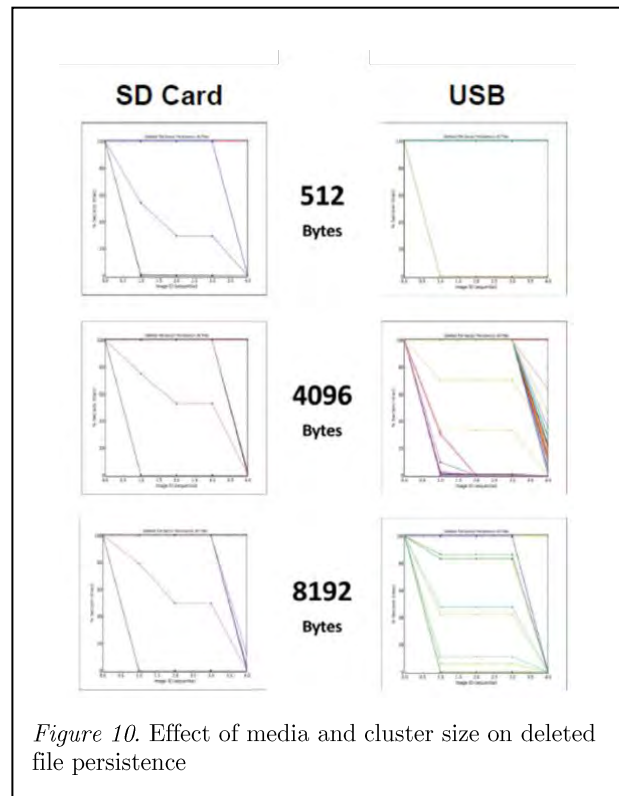


Figure 10. Effect of media and cluster size on deleted file persistence

## 5. CONCLUSIONS AND FUTURE WORK

We present results from multiple experiments exploring the factors affecting deleted file persistence on digital cameras with SD card storage, direct SD card storage, and USB

memory sticks. Our results suggest that influential factors include file sizes of deleted and overwriting files, cluster size especially as it relates to file sizes, media type, and number of overwrites. It is generally accepted that under common conditions of limited overwrites, it is likely that some file fragments may remain indefinitely in the cluster slack space of newly allocated files. A related and possibly non-intuitive result here is that fragments of some files will remain even under conditions of repeated overwrites, as indicated by the exponential decay of Figure 9.

Our conclusions generally match our intuition and the hypotheses posed by others; however, no prior empirical work exists to which our conclusions can be compared. It is our hope and expectation that others will use the tools and methods described here to conduct additional experiments to validate or refute these preliminary results. Our future work will continue to explore the open questions posed by this work. We will also explore additional factors that might affect deleted file persistence on these and other storage devices and systems. Related work currently in progress is exploring deleted file persistence on mobile phones, IoT devices, and industrial control systems equipment.

## ACKNOWLEDGEMENTS

The authors would like to thank MITRE and the National Cybersecurity Center of Excellence for sponsoring the camera experiments described above. The authors would also like to thank the George Mason University Aspiring Scientists Summer Internship Program under which three of the authors participated in this work.

## AUTHOR BIOGRAPHIES

Jim Jones is an Associate Professor in the Electrical and Computer Engineering Department at George Mason University.

Anurag Srivastava is a PhD IT student in the Volgenau School of Engineering at George Mason University. Josh Mosier, Connor Anderson, and Seth Buenafe were high school students and participants in the George Mason University Aspiring Scientists Summer Internship Program at the time of this work.

## REFERENCES

- Bell, G.B. and Boddington, R. (2010) Solid State Drives: The Beginning of the End for Current Practice in Digital Forensic Recovery? *Journal of Digital Forensics, Security and Law*, 5 (3). pp. 1-20.
- Calhoun, W. C., & Coles, D. (2008). Predicting the types of file fragments. *Digital Investigation*, 5, S14-S20.
- Casey, E., & Turnbull, B. (2011). Digital evidence on mobile devices. Eoghan Casey, *Digital Evidence and Computer Crime*. Third Edition. Forensic Science, Computers, and the Internet, Academic Pres.
- Conrad, S., Rodriguez, C., Marberry, C., & Craiger, P. (2009). Forensic Analysis of the Sony PlayStation Portable. In *Advances in digital Forensics V* (pp. 119-129). Springer Berlin Heidelberg.
- Fairbanks, K., & Garfinkel, S. (2012). Column: Factors Affecting Data Decay. *Journal of Digital Forensics, Security and Law*, 7(2), 7-10.
- Farmer, D., & Venema, W. (2005). *Forensic discovery* (Vol. 6). Upper Saddle River: Addison-Wesley.
- Garfinkel, S. L. (2007). Carving contiguous and fragmented files with fast object validation. *Digital Investigation*, 4, 2-12.

- Garfinkel, S. L., & McCarrin, M. (2015). Hash-based carving: Searching media for complete files and file fragments with sector hashing and hashdb. *Digital Investigation*, 14, S95-S105.
- Garfinkel, S., Nelson, A., White, D., & Roussev, V. (2010). Using purpose-built functions and block hashes to enable small block and sub-file forensics. *digital investigation*, 7, S13-S23.
- Garfinkel, S., Nelson, A. J., & Young, J. (2012). A general strategy for differential forensic analysis. *Digital Investigation*, 9, S50-S59.
- Govan, M. (2013, June). Forensic Droplets & Puddles from the Cloud. The 3rd International Conference on Cybercrime, Security and Digital Forensics, Cardiff; June 2013.
- Gubanovs and Afonin (2014). Recovering Evidence from SSD Drives in 2014: Understanding TRIM, Garbage Collection and Exclusions. Online at <http://articles.forensicrofocus.com/2014/09/23/recovering-evidence-from-ssd-drives-in-2014-understanding-trim-garbage-collection-and-exclusions/>
- Huang, N., He, J., Zhao, B., Liu, G., & Wan, X. (2015). Reconstructing Fragmented YAFFS2 Files for Forensic Analysis. *International Journal of Hybrid Information Technology*, 8(7), 37-44.
- Jones, J., & Khan, T. (2017). A Method and Implementation for the Empirical Study of Deleted File Persistence in Digital Devices and Media. Proceedings of the 2017 IEEE 7th Annual Computing and Communication Workshop and Conference (CCWC).
- Joukov, N., Papaxenopoulos, H., & Zadok, E. (2006, October). Secure deletion myths, issues, and solutions. In Proceedings of the second ACM workshop on Storage security and survivability (pp. 61-66). ACM.
- Li, B., Wang, Q., & Luo, J. (2006, December). Forensic analysis of document fragment based on SVM. In *Intelligent Information Hiding and Multimedia Signal Processing, 2006. IHH-MSP'06. International Conference on* (pp. 236-239). IEEE.
- Litchfield, D. (2007). Oracle forensics part 2: Locating dropped objects. NGSSoftware Insight Security Research (NISR) Publication, Next Generation Security Software.
- Nagy, S., Palmer, I., Sundaramurthy, S. C., Ou, X., & Campbell, R. (2015). An Empirical Study on Current Models for Reasoning about Digital Evidence. 10th International Conference on Systematic Approaches to Digital Forensic Engineering. Malaga, Spain.
- Ravi, A., Kumar, T. R., & Mathew, A. R. (2016). A method for carving fragmented document and image files. In 2016 International Conference on Advances in Human Machine Interaction (HMI) (pp. 1-6). IEEE.
- Reust, J., & Friedburg, S. (2006). DFRWS 2005 Workshop Report.
- Roussev, V., & Quates, C. (2012). Content triage with similarity digests: The M57 case study. *Digital Investigation*, 9, S60-S68.
- Roussev, V., & Quates, C. (2013). File fragment encoding classification—An empirical approach. *Digital Investigation*, 10, S69-S77.
- Sencar, H. T., & Memon, N. (2009). Identification and recovery of JPEG files with missing fragments. *digital investigation*, 6, S88-S98.

- Stahlberg, P., Miklau, G., & Levine, B. N. (2007, June). Threats to privacy in the forensic analysis of database systems. In Proceedings of the 2007 ACM SIGMOD international conference on Management of data (pp. 91-102). ACM.
- Uzun, E., & Sencar, H. T. (2015). Carving orphaned JPEG file fragments. *IEEE Transactions on Information Forensics and Security*, 10(8), 1549-1563.
- Yi, S., Hu, X., & Wu, H. (2015). An automatic reassembly model and algorithm of log file fragments based on graph theory. In *Software Engineering and Service Science (ICSESS), 2015 6th IEEE International Conference on* (pp. 686-689). IEEE.
- Zhang, L., Hao, S. G., Zheng, J., Tan, Y. A., Zhang, Q. X., & Li, Y. Z. (2015). Descrambling data on solid-state disks by reverse-engineering the firmware. *Digital Investigation*, 12, 77-87.

