THE JOURNAL OF DIGITAL FORENSICS, SECURITY AND LAW

Journal of Digital Forensics, Security and Law

Volume 12 | Number 4 Article 7

12-2017

Evidence Verification Complications with Solid-State Drives

Ryne Teague
University of South Alabama, rteague2566@gmail.com

Michael Black
University of South Alabama, mblack@southalabama.edu

Recommended Citation

Teague, Ryne and Black, Michael (2017) "Evidence Verification Complications with Solid-State Drives," *Journal of Digital Forensics, Security and Law*: Vol. 12 : No. 4 , Article 7.
DOI: https://doi.org/10.15394/jdfsl.2017.1445
Available at: https://commons.erau.edu/jdfsl/vol12/iss4/7

Evidence Verification Complications with Solid-State Drives

Cover Page Footnote
None

EVIDENCE VERIFICATION COMPLICATIONS WITH SOLID-STATE DRIVES

Ryne Teague, M.S.
Michael Black, Ph.D.
University of South Alabama
Mobile, AL 36688
251-460-6880
mblack@southalabama.edu

ABSTRACT

Solid-state drives operate on a combination of technologies that create a barrier between the physical data being written and the digital forensics investigator. This barrier prevents the application of evidence verification methods developed for magnetic disk drives because the barrier prevents the investigator from directly controlling and therefore verifying that the underlying physical data has not been manipulated. The purpose of this research is to identify a period of inactivity where the underlying physical data is not being manipulated by wear-leveling or garbage-collection routines such that evidence can be reliably verified with existing hashing algorithms. An experiment is conducted on Samsung drives. The limitation of this method is it does not enable the verification of deleted data and will be one size of solid-state drives. The results show that after an hour and a half, the solid-state drives examined will produce the same consistently until ten hours.

Keywords: SSD, verification, validation, chain of custody, solid-state drive

1. INTRODUCTION

Methods for forensics and storage techniques have been developed around the physical design of magnetic hard drives and the idea of evidence verification [1][2]. Digital evidence altered during an investigation can be ruled inadmissible in a court of law. Hard disk drives and older forms of storage have proven techniques to prevent changes to data [1][3][4]. It is also possible to provide evidence that a hard disk drive has not been altered using a hashing program. With these methods, digital evidence can be presented in court with little question of their integrity [2][3].

The design of solid-state drives has drastically changed from the design of hard disk drives and because of this, evidence verification procedures used for magnetic drives will need to be redesigned for use in solid-state drives [1]. A solid-state drive's storage medium has a limited number of writes before a section of the memory will fail, rendering the entire drive useless [1][2]. A component of the solid-state drive is the solid-state drive controller that holds firmware used to increase the lifespan of the drive [1]. The solid-state controller's introduction has led to a wider adoption of solid-state technology, but it

has also complicated standards for digital evidence in forensics investigations [1].

Previous evidence verification methods fail due to certain firmware processes within the solid-state drive controller. Users are not given the same level of control over solid-state drives as they are hard disk drives [1][2]. The mechanism to prevent changes in HDDs works because of the lack of a solid-state controller equivalence inside HDDs drive [4].

The research goal and question were identified by searching for solutions to the problems from solid-state drives that forensic investigators face. The research goal is to explore periods of inactivity in solid-state firmware processes, also known as housekeeping algorithms, that affect evidence verification. The research question is as follows. Can periods of inactivity be identified during which housekeeping algorithms of a solid-state drive's controller does not affect the validation process done by digital forensics investigators using a court accepting hashing methodology?

2. METHOD

The goal of this research is to contribute to digital forensics knowledge base in the areas of solid-state drives and evidence verification by finding solutions without physically altering the solid-state drive. This experiment used the quantitative experimental research approach where a problem had been identified, data was collected, and a hypothesis had been formulated and to be tested. The experiment was designed to observe one condition which was the length of time between the solid-state drive to reach a static state. The results were measured and stored using software and are relevant to the research goals.

The purpose of this study is to identify if there is a period of inactivity where the validation process used by a forensic investigator is not being manipulated by wear-leveling or garbage-collection routines. The test has been conducted as a controlled experiment. A period of inactivity is defined as a time when the solid-state drive is not manipulating the underlying data. Inactivity is verified by hashing the drive and verifying the output. Based on Bell's experiment, any window of inactivity is most likely to be soon after the drive has been powered up therefore the test will only be repeated for two hours.

The researcher used four consumer-rated solid-state drives with a capacity of one hundred and twenty gigabytes to one hundred and twenty-eight gigabytes for this experiment. As stated before, Samsung controls most of the solid-state market. Therefore, all the solid-state drives used were manufactured by Samsung varying in size between one hundred and twenty and one hundred and twenty-eight gigabytes.

An environment was necessary to create a testbed to run consistent experiments on each solid-state drive. The environment consisted of one desktop computer, the necessary input/output devices for the computer, and four solid-state drives. Each solid-state drive was connected to the experiment computer directly through a SATA port. The experiment computer was installed with Ubuntu Server 14.04.05 LTS operating system. After installation, the operating system was immediately patched with up-to-date software packages. The only additional software installed were VIM, a simple text editor and parted, a command line tool used to partition drives. A Linux operating system was chosen over Windows because Windows modifies system volume information on drives utilizing NTFS file systems which may issue write commands to the experiment drives. No such requirement exists on the Ubuntu operating system. During the experiments, data sent to the test drives was controlled by scripts.

The experiment began by connecting each solid-state drive tested to the experiment machine. The machine was then powered on and booted to the Ubuntu operating system. Each solid-state drive had both data and power cables disconnected and reconnected. This was done to remove the frozen status of the solid-state drives. Upon booting, an operating system issues an ATA Freeze command that prevents any security-related changes to the solid-state drives. Power cycling the test drives unfroze each drive and allowed the secure-erase command to operate properly.

Four terminals were opened to execute the experiment scripts on each of the test drives. Each terminal was set to a different directory where the experiment script was copied. Each experiment script was then started and passed the path to a solid-state drive and the location of a unique mount point as parameters. The syntax to run the script at the command line is the following:

./main_1.sh <drive> <mountpoint>

The script began by issuing the secure-erase command erasing all contents the drive. The script then partitioned, formatted, and mounted the solid-state drives. By writing standard text files, the solid-state drives were then filled to ninety percent capacity, deleted to twenty-five percent capacity, and then filled to fifty-percent capacity. Upon completion of all file write and deletion, an MD5 hashing program was immediately executed for two hours on each solid-state drive.

It should be noted that the solid-state drives never reached one-hundred percent capacity. It was considered rare in a real-world scenario that a user's drive would ever reach maximum capacity. The deletion of data created slack space which should cause garbage-collection and wear-leveling to be triggered. The process of writing, deleting, and writing files again was intended to ensure every available cell of the test drive to be written to at least once and to have cells that are marked for deletion. Together, these scripts created a scenario that would trigger garbage-collection and wear-leveling routines to activate.

In total, the experiment was conducted four times. In the first two experiments, the MD5 hashing program executed for two hours. In the last two runs the MD5 hashing program executed for ten hours. The time for one experiment was originally estimated to take between two to three hours for each solid-state drive, totaling between ten to fifteen hours for all. The actual time for completion of four tests executed on four drives concurrently took between twenty-seven to twenty-nine hours.

The detailed experiment steps are below. This experiment has eleven steps. Steps three through ten were implemented in scripts developed by the researcher. All scripts were run from the experimenter's machine.

1. Experimenter's machine was powered on with four solid-state drives connected plus an additional magnetic disk drive that hosts Ubuntu 14.04.05 lts operating system.
2. The experimenter disconnected the power and data cable from each solid-state drive from the machine and then reconnects both cables again.
3. Each solid-state drive was wiped by being sent a Secure-Erase ATA command which defaulted the drive to its original, out of the box state.
4. The script then paused until each drive completed the Secure-Erase.
5. Each solid-state drive was then formatted with the NTFS file system.
6. Each solid-state drive was logically mounted to the experimenter's system.
7. Each solid-state drive was then written to ninety percent capacity.

8. The drives were written with files ranging from one byte to thirty-two megabytes in size from /dev/urandom.
9. Each solid-state drive had randomly selected files deleted until the drive reached twenty-five percent capacity.
10. The deletion operation was a normal file-system deletion, like emptying the recycle bin of a windows operating system.
11. Each solid-state drive was then written to fifty percent capacity.
12. The drives were written with files ranging from one byte to thirty-two megabytes in size from /dev/urandom.
13. A script was run from the researcher's computer to produce an MD5 of the solid-state drive continuously for two hours. Each MD5 produced was stored on the computer for analysis along with the time taken to generate the MD5.
14. Data was analyzed to identify periods of inactivity within the solid-state drive's garbage-collection and wear-leveling processes.

3. RESULTS

The results are formatted in tables with the MD5 hash in the left column and the time taken to produce the MD5 hash in the right column. The time is formatted as minutes:seconds:milliseconds.

Table 1.
Drive A 2 Hour MD5 Hash Output, Experiment 1

#	Hash	Duration
1	bac32beed77d6d8abf8638fc9a22134c	25:35.52
2	b358d9a12daaa67085f1c3b7f5bc79c8	24:43.11
3	b358d9a12daaa67085f1c3b7f5bc79c8	24:46.05
4	b358d9a12daaa67085f1c3b7f5bc79c8	22:43.24
5	b358d9a12daaa67085f1c3b7f5bc79c8	21:33.29
6	b358d9a12daaa67085f1c3b7f5bc79c8	15:53.90

Table 2.
Drive B 2 Hour MD5 Hash Output, 2, Experiment 1

#	Hash	Duration
1	81e5f26cf86a339b640891344d252111	20:38.00
2	14478f9c87d07333e2b094c1ae6b8e43	21:28.35
3	7d7518901d13edc55bc7faf53c3dc1d4	22:56.48
4	7d7518901d13edc55bc7faf53c3dc1d4	23:39.49
5	7d7518901d13edc55bc7faf53c3dc1d4	23:06.82
6	7d7518901d13edc55bc7faf53c3dc1d4	23:02.83

Table 3.
Drive C 2 Hour MD5 Hash Output, Experiment 1

#	Hash	Duration
1	f6c0843420e90891d2c38ef218920a56	26:20.99
2	a5187d990a919bfdec2004150fc7b71c	24:55.80
3	a5187d990a919bfdec2004150fc7b71c	25:19.84
4	a5187d990a919bfdec2004150fc7b71c	22:53.60
5	a5187d990a919bfdec2004150fc7b71c	21:48.87

Table 4.
Drive D 2 Hour MD5 Hash Output, Experiment 1

#	Hash	Duration
1	f390b733f0e18d5c0ba9261e6d63475d	10:20.35
2	e6dbfc25b51fbd8cb0205b8303d3583c	10:41.85
3	e6dbfc25b51fbd8cb0205b8303d3583c	10:49.51
4	5369c891d3abe93586646338c056103e	11:07.67
5	5369c891d3abe93586646338c056103e	11:08.67
6	5369c891d3abe93586646338c056103e	12:18.61
7	5369c891d3abe93586646338c056103e	12:28.77
8	5369c891d3abe93586646338c056103e	11:38.91
9	5369c891d3abe93586646338c056103e	11:40.61
10	5369c891d3abe93586646338c056103e	11:33.02
11	5369c891d3abe93586646338c056103e	12:23.23

As shown from the first experiment's results, the hash became consistent after the third consecutive hash. When the same hash is produced consecutively, the contents of the drive are not being altered. These results imply there are periods of inactivity in the solid-state drive controller's housekeeping algorithms during which validation processes are not affected.

Except for one solid-state drive, each drive reached a state where the hashes were not changing after the MD5 hash program executed once. The experiment was repeated a second time to validate the results of the first experiment.

Table 5.
Drive A 2 Hour MD5 Hash Output, Experiment 2

#	Hash	Duration
1	856002ae8684fc21f1bf5c82ef568128	20:38.97
2	ed8e45e37d6a81aab0ac441e88187c45	22:47.51
3	ed8e45e37d6a81aab0ac441e88187c45	22:22.07
4	27764daaea24b2877db98fed10225c95	22:36.22
5	27764daaea24b2877db98fed10225c95	22:02.87
6	27764daaea24b2877db98fed10225c95	21:47.60

Table 6.
Drive B 2 Hour MD5 Hash Output, Experiment 2

#	Hash	Duration
1	e189f3a8db0bfaf22fd4debcd36ee454	15:57.91
2	4fef78663fb013d43aa68070fd3f5c2c	15:52.62
3	4fef78663fb013d43aa68070fd3f5c2c	17:43.63
4	4fef78663fb013d43aa68070fd3f5c2c	21:02.38
5	4fef78663fb013d43aa68070fd3f5c2c	21:20.30
6	4fef78663fb013d43aa68070fd3f5c2c	20:56.44
7	b56ee7e5e45b25011da4508be4d296ac	20:46.62

Table 7.
Drive C 2 Hour MD5 Hash Output, Experiment 2

#	Hash	Duration
1	585740b343ff7d932c15577986fd70ce	21:46.95
2	c5605404931fc7de248495a87127476f	23:10.35
3	c5605404931fc7de248495a87127476f	22:45.43
4	9d1fccb5c68ccf2c2d98265372cf944f	22:52.67
5	9d1fccb5c68ccf2c2d98265372cf944f	22:26.83
6	9d1fccb5c68ccf2c2d98265372cf944f	22:47.63

Table 8.
Drive D 2 Hour MD5 Hash Output, Experiment 2

#	Hash	Duration
1	d7c6dd4c527b2339c74cf07a6a1cfed6	15:53.15
2	6fd19891def04f0b9b009e88edcb736c	15:53.49
3	6fd19891def04f0b9b009e88edcb736c	17:40.12
4	6fd19891def04f0b9b009e88edcb736c	20:55.82
5	6fd19891def04f0b9b009e88edcb736c	21:18.84
6	6fd19891def04f0b9b009e88edcb736c	20:56.35
7	bdecd52de5aaff448471cb1aef123455	20:48.63

In the second experiment, Drive A and Drive C produced a consistent hash after four hashing iterations. These results match the first experiment's results. However, the results for Drive B and Drive D were different from the results in the first experiment, and Drives A and C in the second experiment.

The last hash produced for Drive B and D was different from earlier hashes which could indicate that garbage-collection and wear-leveling routines were running again after a period of inactivity. Conclusions were drawn from these results that the wear-leveling and/or garbage-collection could still run after entering a state of inactivity. Additionally, Drive A and Drive C only produced a consistent hash for three hashing iterations before the test ended. Drive B and Drive D had produced consistent consecutive hashes for longer but still allowed data to be altered at the end of the test. This may be an indication that Drive A and Drive C's wear-leveling and or garbage-collection could also activate after a period of inactivity.

Additional data was gathered to examine how often the hash changed for each drive. A second set of tests was conducted the same as the first tests, with the exception that the MD5 hashing program would execute for an additional eight hours changing the total execution time to ten hours.

Table 9.
Drive A 10 Hour MD5 Hash Output, Experiment 3

#	Hash	Duration
1	508267f0208523d00ba7f0372966fb79	19:57.9
2	59a78271761fb9f36b30fd68efe6a860	20:30.0
3	1f84ca40ac124b71a57d021fa12a9e90	21:34.0
4	1910b0ffd7435d4d82869961efef7360	22:07.5
⋮	⋮	⋮
28	1910b0ffd7435d4d82869961efef7360	20:25.9

Table 10.
Drive B 10 Hour MD5 Hash Output, Experiment 3

#	Hash	Duration
1	63b38a64b2ecaed4949d8db6320e9f48	15:24.1
2	9b9b0ddfe1f703d7c465c3539feb04ed	15:40.4
3	9b9b0ddfe1f703d7c465c3539feb04ed	17:40.8
4	9b9b0ddfe1f703d7c465c3539feb04ed	19:24.2
5	9b9b0ddfe1f703d7c465c3539feb04ed	19:13.8
6	d72a2e1718d8a89c79da3e31ac8ab47e	21:07.5
⋮	⋮	⋮
32	d72a2e1718d8a89c79da3e31ac8ab47	21:05.3

Table 11.
Drive C 10 Hour MD5 Hash Output, Experiment 3

#	Hash	Duration
1	0507b030bc15b96fe50f353efa3e2c55	20:40.1
2	f730256327f8f4a1f9e3e754aa36b32d	20:53.4
3	0f1765678198327ad6327703dd953d70	22:25.2
⋮	⋮	⋮
26	0f1765678198327ad6327703dd953d70	23:33.9

Table 12.
Drive D 10 Hour MD5 Hash Output, Experiment 3

#	Hash	Duration
1	1e2d732187a3ec427efcd3422fe1ad7b	15:13.6
2	8cca98cc1a5634b7f033682d5b8f3e6a	15:37.6
3	8cca98cc1a5634b7f033682d5b8f3e6a	17:17.6
4	8cca98cc1a5634b7f033682d5b8f3e6a	19:23.2
5	8cca98cc1a5634b7f033682d5b8f3e6a	18:56.6
6	d8ff920d43933fe2aef94fa532de5a3a	20:56.7
⋮	⋮	⋮
30	d8ff920d43933fe2aef94fa532de5a3a	20:40.1

Table 13.
Drive A 10 Hour MD5 Hash Output, Experiment 4

#	Hash	Duration
1	6a5a93c167f797bd46a80de292fb324a	22:45.9
2	8db0034679607dfa8514444f2a01bfce	23:44.0
⋮	⋮	⋮
26	8db0034679607dfa8514444f2a01bfce	25:32.6

Table 14.
Drive B 10 Hour MD5 Hash Output, Experiment 4

#	Hash	Duration
1	ee94482f39bca3f9244d9e172bbccf7c	15:47.6
2	b3cc0f055a8b347e6b42fcff2f70113e	15:48.0
⋮	⋮	⋮
28	b3cc0f055a8b347e6b42fcff2f70113e	23:22.2

Table 15.
Drive C 10 Hour MD5 Hash Output, Experiment 4

#	Hash	Duration
1	f3bbdad1755465de46266fc33a92713a	12:23.1
2	95a054dae1826fa8e01bff9ed45f0aa5	11:55.8
⋮	⋮	⋮
49	95a054dae1826fa8e01bff9ed45f0aa5	14:52.9

Table 16.
Drive D 10 Hour MD5 Hash Output, Experiment 4

#	Hash	Duration
1	ff49c215d6b597dbb6fde20ce6353357	15:42.2
2	7705414505849254d18e29d096f12980	15:49.9
⋮	⋮	⋮
49	95a054dae1826fa8e01bff9ed45f0aa5	14:52.9

4. CONCLUSION

Based on the results seen in Table 8 through Table 16, all solid-state drives tested produce a consistent hash when the experiment was extended from two to ten hours. Additionally, seven of the sixteen tests show an early period of consecutive, matching hashes may occur within minutes of the start of the analysis of the drives but change again within several hashing iterations.

The research question is "can periods of inactivity be identified during which housekeeping algorithms of a solid-State drive's controller does not affect the validation process done by digital forensics investigators using a court accepting hashing methodology?" The data produced by the experiment above provides a strong indication that given an amount of time equal to or greater than one and a half hours, a solid-state drive will reach a state in which multiple matching hashes are generated and where the validation process for digital forensic investigators is not affected using these solid-state drives and firmware versions.

These results have also produced data about the aggressiveness of the garbage-collection and wear-leveling routines of solid-state drives. The only variable changed during the experiments was the length of time the MD5 hashing program executed. However, the aggression and speed of the wear-leveling and garbage-collection routines are unpredictable in these experiments. Experiment three for Drive A showed the MD5 hashing program produced different hash five iterations in a row until reaching a state where consecutive matching hashes were produced, which only happened in one experiment. Drive A reached a state where consecutive matching hashes were produced on experiment three after one hour but reached the same state in experiment four after only twenty-two minutes. Additionally, Drive B showed a short, early consecutive matching hashes state in experiment three, but not in experiment four.

The results indicate that there is a period of inactivity where the solid-state drive produces a consistent hash over a length of time that could allow an investigator to image the solid-state drive, verify the image, and power off the solid-state drive before the hash changes. However, allowing the drive to reach this period of inactivity before a forensic image is created causes data such as slack space to be lost and changes the drive from its original acquisition state. Presently, that is not within accepted forensic practices. New forensic practices should be reviewed to adopt this technique. If new chain of custody standards approved of this technique, investigators could use this technique to obtain evidence from a forensic copy that is close to the solid-state drive's original state. Currently, evidence obtained from solid-state drives is usually deemed inadmissible because solid-state drives are self-altering and self-corroding due to wear-leveling routines and garbage-collection.

When this research began, the goal was to investigate whether solid-state drives can reach a state that no longer interferes with the validation process used by forensic examiners. The research has identified a strong indication to support this hypothesis. The experiment wrote data mimicking a user's daily use in real environments, and then examined four solid-state drives produced by the industry's leading producer of drives. The results of this experiment clearly show that a period of inactivity for garbage-collection and wear-leveling routines was identified within the ten-hour window; however, this period of inactivity has not yet been proven to be permanent. Further research and analysis is needed on this subject before a method can be published that allows forensic evidence to be examined and presented in court without question.

REFERENCES

[1] Bell, G. B., & Boddington, R. (2010). Solid State Drives: The Beginning of the End for Current Practice in Digital Forensic Recovery? The Journal of Digital Forensics, Security and Law, 5(3), 1–20.

[2] Burd, S. (2011). Systems architecture (6th ed.). Boston, Massachusetts: Cengage Course Technology.

[3] Carrier, B. (2005). File System Forensic Analysis. Upper Saddle River, New Jersey: Pearson Education.

[4] Volonino, L., Anzaldua, R., & Godwin, J. (2007). Computer Forensics Principles and Practices. Upper Saddle River, New Jersey: Pearson Education.

APPENDIX

Drive A
Model: SAMSUNG MZ7TD128HAFV-000L1
Serial Number: S14TNSAD602536
Firmware Revision: DXT04L0Q

Drive B
Model: Samsung SSD 750 EVO 120GB
Serial Number: S33MNB0H949501E
Firmware Revision: MAT01B6Q

Drive C
Model: SAMSUNG SSD 830 Series
Serial Number: S0VUNYABA01720
Firmware Revision: CXM01B1Q

Drive D
Model: Samsung SSD 840 EVO 120GB
Serial Number: S1D5NSAFB67977M
Firmware Revision: EXT0CB6Q

