Classifying Proprietary Firmware on a Solid State Drive Using Idle State Current Draw Measurements

T. OWENS WALKER, III1, (Senior Member, IEEE), JUSTIN A. BLANCO1, (Senior Member, IEEE), RYAN RAKVIC1, (Senior Member, IEEE), ANN VANLEER1, DANE BROWN2, JAMES SHEY1, GREGORY L. SINSLEY1, HAU T. NGO1, AND ROBERT W. IVES1, (Senior Member, IEEE)

1Electrical and Computer Engineering Department, United States Naval Academy, Annapolis, MD 21402-5025, USA
2Cyber Science Department, United States Naval Academy, Annapolis, MD 21402-5025, USA

ABSTRACT Solid state drives (SSDs) are coming under increased scrutiny as their popularity continues to grow. SSDs differ from their hard disk drive predecessors because they include an onboard layer of firmware to perform required maintenance tasks related to data location mapping, write performance, and drive lifetime management. This firmware layer is transparent to the user and can be difficult to characterize despite its clear potential to impact drive behavior. Flaws and vulnerabilities in this firmware layer have become increasingly common. In this work, we propose and analyze a technique to classify different versions of proprietary firmware on an SSD through the use of current draw measurements. We demonstrate that major groupings of firmware can be classified using current draw measurements not only from explicitly active drive states such as read and write but also from the low power idle state. We achieve pairwise classifications rates near 100% between firmware examples in these different major groupings. Coupling these results with firmware release information, we are able to infer major updates in the firmware timeline for the SSD we examined. We also develop an anomaly detector and achieve detection rates of 100% for samples that reside outside of the reference grouping.

INDEX TERMS Solid state drive, SSD, current draw, power consumption, firmware, security.

I. INTRODUCTION

Flaws and vulnerabilities continue to plague the growing solid state drive (SSD) market. In 2009, Intel halted shipments on its X25M and X-18M SSDs due to the presence of a bug that corrupted user data [1] and researchers warned of vulnerabilities present in manufacturer supply chains [2]. In 2013, KingFast inadvertently shipped a counterfeit SSD with fake NAND memory to a reviewer [3]. More recently in 2019, researchers found flaws in Crucial and Samsung SSDs that allow the encryption to be bypassed [4] and Intel was forced to release a patch to correct a privilege escalation vulnerability in some of its enterprise SSDs [5].

While bugs, vulnerabilities, and malware are nothing new in the cyber security field, they can be difficult to detect when they reside in the firmware of a commercial SSD. Users have little visibility on the functionality of this proprietary firmware that is needed to map logical memory to physical flash memory and limit physical wear on the transistors. To expose these hidden vulnerabilities, our group has recently demonstrated the use of current draw measurements to provide insight into this functionality [6]–[9]. In the security context of malware detection, forensic analysis, and consumer protection, Shey et al. [6] developed an automatic classifier capable of identifying the clearing of physical data locations while Brown et al. [7] has proposed the use of these measurements to detect firmware modification in open-source SSDs. It has also been shown that current draw measurements can
be used to infer the resident file system type [8] as well as the presence of read and write operations [9].

Leveraging this prior work on side-channel current draw analysis, the primary contributions of this article are as follows:

1) Demonstrates a technique for classifying firmware revisions in a family of proprietary firmware versions on a commercial solid state drive using non-intrusive current draw measurements.

2) Demonstrates the use of this classification technique coupled with firmware release information to infer major firmware revisions in the examined firmware timeline.

3) Demonstrates that classification can be accomplished using current draw measurements not only from explicitly active drive states (i.e., read and write) but also from the low power idle state.

4) Develops an anomaly detector designed to identify firmware versions that differ from a reference firmware or firmware grouping.

To our knowledge, this is the first examination of side channel leakage for a family of proprietary firmware versions using power-related metrics and the first work to perform classification based solely on idle state current draw measurements. In practical application, the techniques provided here would allow a researcher or security analyst to infer whether or not the firmware installed on the SSD was a member of the reference firmware grouping without the need to reverse engineer it.

This article is organized as follows. Related work is discussed in Section II while Section III details the relevant firmware versions and identifies the four observed modes of operation for the firmware. The experimental setup is described in Section IV and the accompanying analysis is presented in Section V. Section VI presents the results and the article finishes with conclusions and future work in Section VII.

II. RELATED WORK

As their popularity grows, the security of SSDs has come under increased scrutiny by researchers in the past several years. In this section, we review the relevant existing literature. We begin with a brief discussion of other firmware-focused security research being conducted on SSDs and then provide a discussion of power-related side channel analysis both as it relates to computer systems and processors in general as well as SSDs specifically.

Like their hard disk drive predecessors [10], the firmware on SSDs is vulnerable to compromise regardless of whether it is open- or closed-source. Representative of the former, Bogaard and Bruijn demonstrated the feasibility of inserting a functioning backdoor into open-source SSD firmware [11] but recognized that this would be significantly more difficult on proprietary firmware where access was restricted by the manufacturer. This latter challenge has recently been overcome by Meijer and Gastel who demonstrated the ability to compromise proprietary, closed-source firmware on both Crucial and Samsung SSDs using techniques that included physical access and code injection [4]. In other related work, the firmware of SSDs has also been modified to detect and recover from ransomware attacks [12]. Our work builds on these vulnerability studies by proposing and analyzing a technique to assist investigators in distinguishing between different versions of firmware that may be resident on a commercial SSD. This would be particularly useful in scenarios in which the resident firmware is not known or the SSD is suspected to have been tampered with.

Power-related measurements including current draw and associated side-channel attacks have been used extensively in the context of both security and performance. Of note, power analysis has been used to characterize as well as disassemble processor instructions [13], [14], detect malware on programmable logic controllers [15], and classify attacks on a Raspberry Pi [16]. It has also been used as part of larger side-channel analysis-based toolboxes such as SASEBO [17] and RamDPA [18]. In direct application to SSDs, current draw measurements have been used to characterize write performance on the Intel X25M and the Samsung MXP SSD families [19].

The work presented here follows a recent string of advancements in the use of current draw to infer SSD operations and characterize SSD parameters. Current draw measurements have been used to infer TRIM operations [6] as well as read and write operations [9]. They have also been used to characterize the file system in use on the SSD [20] and the encryption algorithm present [21]. Most recently, current draw measurements have been proposed to detect modifications of firmware on an open-source SSD [7]. While this work evolved from these earlier findings, it is the first to demonstrate the ability to characterize SSDs using the low power idle state and it is the first to classify a family of proprietary, closed-source firmware versions on a commercial, off-the-shelf SSD.

This article builds upon a set of initial findings presented in [22]. This earlier work demonstrated the ability to distinguish between two firmware versions using read and write operations on files of different sizes. In the current work, we have demonstrated the ability to classify firmware versions using the idle state only and have extended the analysis to the family of firmware versions available for the SSD under examination. This has led to important insights, particularly in terms of recognizing major updates within this examined firmware family. We also assess the potential effectiveness of an anomaly detector based on this technique.

III. FIRMWARE SPECIFICATIONS AND OPERATING MODES

The Crucial m4 SSD, released in 2011, was selected for this work because a substantial number of firmware release versions are publicly available for it and the associated firmware upgrade utility provides the ability to both upgrade
and downgrade the firmware. (Specifications for the Crucial m4 SSD utilized in this work are provided in Table 1.) In this section, we provide an overview of the firmware versions available for the Crucial m4 and discuss the four observed operating modes.

A. Firmware Specifications

Seven firmware revisions have been released for the Crucial m4 SSD: 0002 on 8 June 2011 [23], 0009 on 25 August 2011 [23], 0309 on 13 January 2012 [24], [25], 000F on 5 April 2012 [26], [27], 010G on 25 September 2012 [28], [29], 040H on 4 December 2012 [30], [31], and 070H on 2 April 2013 [32], [33]. This firmware timeline is summarized in Fig. 1. We have also indicated the Windows 8 release date in this figure because, as will be seen later in this work, there is a clear delineation before and after Windows 8 enhancements were introduced in firmware version 010G.

TABLE 1. Crucial m4 SSD specifications.

| Model          | Specification   |
|----------------|-----------------|
| Capacity       | 64 GB           |
| Interface      | SATA 6 Gb/s     |
| Read speed     | 500 MB/s        |
| Write speed    | 95 MB/s         |
| Manufacturer-installed firmware | Version 0009 |

The first two firmware updates (0002 and 0009) were released prior to a subsequent fix in update 0309 that corrected a “blue screen of death” error that occurred after 5,184 hours of “on time” [34]. We begin our work with version 0309. Version 000F followed 0309 and introduced some modest performance improvements.

Version 010G represented an important milestone in the update history of the m4 firmware family as it was the first update to provide a number of performance enhancements targeted specifically at Windows 8 systems. These improvements were apparently based on Windows 8 pre-release versions made available to Crucial as the official version of the new operating system was released about a month after firmware version 010G (as shown in Fig. 1). Firmware versions 040H and 070H (the latter is the current version of the firmware) followed shortly afterward and largely addressed performance-related issues associated with power-up and recovery.

Additional details for each of the seven firmware versions are included in the Appendix.

B. Observed Modes of Operation

In this work, four distinct modes of SSD operation were observed and investigated: (1) an off state, (2) a low power idle state, (3) a read state, and (4) a write state. The off state is essentially defined as the state in which no current draw from the SSD is present. In practice, it was observed that this off state can be entered by either turning the host computer off or putting the host computer in its low power sleep state. For the results presented in this work, the off state was entered by putting the host computer to sleep. The read state or write state are entered by issuing read or write file commands through the operating system, respectively, from the host computer. We define the idle state as the state in which the SSD is not in one of the other three states (off, read, or write). This is consistent with literature from Crucial which
IV. EXPERIMENTAL SETUP
A. EQUIPMENT
The testbed for this work is comprised of a host computer, a current probe with accompanying amplifier, a data recorder, and the SSD under test. The specifications for the host computer, the current probe, and the data recorder are provided in Tables 2, 3, and 4, respectively. The testbed setup is shown in Fig. 2. A current probe was selected as the primary measurement device because it has been demonstrated to be as effective as an intrusive, in-line resistor installation in related applications [6].

B. DATA COLLECTION
In this subsection, we discuss the data collection techniques used to gather current draw data on the SSD in each of the four operating states: idle, off, read, and write. In all cases, the current probe was attached to the 5V SATA power line and the Gen3i Data Recorder was used to collect samples at a rate of 200 kSamples/sec (as noted in Table 4). The accompanying Perception software suite was used to configure the Gen3i as well as convert and save the data in MATLAB readable *.mat files. In each operating state, ten files were collected for each firmware version based on a predetermined, randomly-generated, collection order. Firmware versions were manually installed using the Firmware Update Guides provided by Crucial [25], [27], [29], [31], [33].

The off state data were collected as follows. The appropriate firmware was installed onto the SSD and the host computer was rebooted. After start-up and login was complete, the computer was put into sleep mode. The off state was visually verified using the data recorder and then 7 seconds of data were recorded. Off data files were approximately 55 MB in size.

To collect the idle state data, the appropriate firmware was installed and the computer rebooted, as before. After start-up was complete, the idle state was visually verified on the data recorder and then 7 seconds of data were recorded. Idle data files were approximately 55 MB in size.

The read and write samples were collected based on the techniques and accompanying Python script developed in [22]. For the write data, a set of files containing random data were generated and written to the SSD. Triggers from the host computer to the data recorder were used to start and stop the data recording before and after the write operation. The read samples were similarly collected by reading randomly populated files that had been previously written to the SSD. In both cases, the computer’s main cache was cleared between each collection. Read and write data files varied in size but were typically 150 MB ± 30 MB. Associated collection times were approximately 20 to 30 seconds.

Representative time domain and spectral plots are provided for each operating state in Fig. 3. These plots illustrate the typical “signature” for each operating state. While it is clear from Fig. 3 that all four operating states differ significantly from each other in both the time and frequency domains, it is important to note that there is also some variability between samples within the same operating state. This can be seen in Fig. 4 which plots ten individual samples of each operating state in the time domain (for firmware version 070H). These differences will also be reflected in the corresponding spectral plots.

We can compare the current draw among the four states in Fig. 5 which plots root mean square (RMS) amplitudes as a function of firmware version for each state. These RMS amplitudes for each of the four distinct modes of SSD operation were calculated from a 2.5 second segment during the operational state for each recorded file. The “active” nature of the read and write states can be clearly seen when contrasted with the low power idle state with an amplitude of less than half that of the active states. Variability among

![FIGURE 2. Testbed used to collect current draw measurements from the Crucial m4 SSD.](image-url)
the different firmware versions for each individual operating state can also be seen in Fig. 5. This variability in RMS amplitudes will also manifest itself in the frequency domain and will be captured in the frequency domain analysis and results presented in the remainder of this work.

V. ANALYSIS
In this section, we discuss the analysis techniques we employ on the data collected during this work. The section opens by describing the features of the SSD current draw signal that are used to distinguish between firmware version groupings and closes with a discussion of the classification algorithms utilized.

A. FREQUENCY DOMAIN FEATURE GENERATION
For each file, 40 raw observations were extracted by dividing seconds 2-6 of the recorded SSD current into 100 ms non-overlapping segments. Since the assumed current draw signatures characterizing each drive state were not always present during the entire recording period (see e.g., Fig. 4b), a fixed analysis window was used to eliminate the need to develop state-dependent current signature detectors prior to feature extraction. The particular 4s window we used was chosen after examining one held out recording from each of the four drive states, as a compromise that balanced the desire for a large statistical sample against computation time for model training and testing while simultaneously capturing characteristic time-domain activity for each state.

Each 100 ms observation was transformed to the frequency domain using Welch’s modified periodogram method [36] (40 ms Hamming-windowed sub-segments with 20 ms of overlap; 8192-point discrete Fourier transforms) to estimate the power spectral density (PSD). The PSD estimate was integrated in non-overlapping frequency bins of 400 Hz width, spanning 0 to 100 kHz, and converted to the decibel scale, yielding a $1 \times 250$ feature vector for each 40 ms observation.

B. CLASSIFICATION EXPERIMENTS
For each of the four SSD operating states (read, write, idle, and off), we conducted two sets of classification experiments. In the first set, we trained binary classifiers to discriminate between all 10 (5 choose 2) possible pairs of firmwares. In the second set, we trained classifiers to discriminate between two firmware groupings {0309, 000F} and {010G, 040H, 070H}. In all experiments, we compared the accuracies of three different supervised learning methods. The first method was logistic regression (LR), in which a linear class boundary is learned by assuming sigmoid models for the posterior probabilities of the classes and finding the model parameters that maximize the conditional likelihood function [37]. The second method was quadratic discriminant analysis (QDA) [37], which is capable of learning a curved class boundary (a quadratic surface). In this method, test observations are assigned to the class with largest posterior probability assuming Gaussian class-conditional densities, and model parameters are estimated via maximum likelihood. The third method of classification was the k nearest neighbors (k-NN)
algorithm [37], an in general still more flexible nonlinear classifier that assigns class labels to test points according to a plurality vote among the closest training points in feature space. For k-NN, we used Euclidean distance as the measure of proximity and 3 as the number of neighbors \( k = 3 \). These three classification approaches were selected because they are representative state-of-the-practice methods [37] that span the spectrum of model flexibility from linear methods with potentially high bias but low variance to highly non-linear methods with potentially low bias but high variance, and hence provide an opportunity to optimize this tradeoff over the considered models to improve generalization performance.

Test set accuracy was estimated using 10-fold (for experiment set 1) or 20-fold (for experiment set 2), leave-one-file-set out cross validation, as described in [6]. In experiment 2, this was achieved by using the 20 available files for the first firmware grouping (10 from 0309 and 10 from 000F) in conjunction with a randomly selected 7, 7, and 6 files, respectively, from firmwares 040H, 070H, and 010G, to give a total of 20 files in the second firmware grouping. Prior to classifier training, inside the cross-validation loop, principal components analysis (PCA) was applied to reduce the dimensionality of the feature set. Specifically, we retained the minimum number of principal components required to account for at least 90% of the data variance. The reduced feature set was used in training, and test data were mapped to the principal component space of the training data before they were classified.

VI. RESULTS

This section presents the current draw-based classification results for the proprietary firmware released for the Crucial m4 SSD. It begins by examining pairwise classification results across the five firmware versions from which two distinct groupings can be seen to emerge. The first grouping includes firmware versions 0309 and 000F while the second grouping includes firmware versions 010G, 040H, and 070H. As noted in the firmware version discussion in Section III, these groupings point to the major firmware update 010G that introduced Windows 8 performance enhancements to the firmware family and pairwise classification based on these groupings is presented. We then demonstrate that it is possible to build an anomaly detector for applications in which trusted examples are available from a reference firmware group and
we confirm that the anomaly detector exhibits the behavior that would be expected based on our binary classification results. The section concludes by comparing classification performance based on measurements taken in the idle state to those taken in the read, write, and off states.

A. PAIRWISE CLASSIFICATION BY INDIVIDUAL FIRMWARE

This section presents the pairwise classification results for all five firmware versions. For comparison, a baseline no information classifier (null) result is provided in each case by randomly permuting the data set labels. Error bars of $\pm$ two standard errors are included to provide an approximate 95% confidence interval around the mean classification accuracy over cross-validation folds. As can be see in Figs. 6 and 7, all three classifiers (logistic regression (LR), quadratic discriminant analysis (QDA), and 3-nearest neighbors (3-NN)) perform similarly across all pairwise sets.

In Fig. 6, it can be seen that the classification is near 100% for all firmware pairs that span across the two firmware groupings (e.g., {0309, 000F} and {010G, 040H, 070H}). However, in Fig. 7, we see that classification is close to chance when we try to distinguish between firmware versions within the same grouping. These pairwise results are summarized in Fig. 8 for the QDA classifier in which the dividing line between the two firmware groupings can be clearly seen. While not shown, similar plots can also be generated for the LR and 3-NN classifiers.

B. PAIRWISE CLASSIFICATION BY FIRMWARE GROUPINGS

Fig. 9 shows the result of a post-hoc analysis in which we group the firmware into two major groups based on whether they were released before or after the Windows 8 enhancements were introduced in 010G. It can be seen that, as expected, classifying across groups yields a classification rate of 100% while classifying within a group is much closer to chance.

C. IDENTIFYING MAJOR UPDATES IN THE FIRMWARE FAMILY

The results of the previous section provide important insight into the manufacturer-provided, closed-source firmware revision timeline. In the case examined here, it appears from Fig. 8 that firmware version 010G potentially represents a major update in the Crucial m4 firmware family. This conclusion arises from the ability to distinguish firmware version pairs that cross this major update boundary (i.e., 0309 vs. 040H and 000F vs. 010G) but not pairs that are strictly on one side or the other of the update boundary (i.e., 0309 vs. 000F and 010G vs. 070H). We can investigate this conclusion further by examining the firmware updates in more detail. Referencing the revision notes provided by Crucial and discussed in Section III, version 010G is the first firmware version to introduce performance enhancements specifically targeted at the Windows 8 operating system.
In addition, Fig. 10 plots the size of the firmware image for each of the different firmware versions. It can be clearly seen that there is a significant increase in firmware image size between 000F and 010G while 000F and 0309 (before the major update) and 010G, 040H, and 070H (after the major update) are relatively close in size to each other, respectively. Given that major firmware updates, which typically result in additional code, are often synchronized with new operating system releases, this aligns with our findings based on the current measurements and indicate that firmware version 010G represents a major update in the firmware timeline.
D. ANOMALY DETECTION

In this section, we demonstrate that we can design an anomaly detector and confirm that it behaves as expected. Before presenting and discussing our results, we begin by describing how they were generated. In addition to the binary classification experiments described in Section V-B, for the idle state, we also trained Gaussian kernel one-class support vector machines [38] for use as anomaly detectors. We used...
TABLE 5. Anomaly detection rates across individual firmware pairs. Dashed lines delineate the boundary between the two firmware groupings.

| Anomaly class | Reference class |
|---------------|----------------|
| 0309          | X              |
| 000F          | 0.270          | 1.000          |
| 010G          | 0.998          | 1.000          |
| 040H          | 1.000          | 0.408          |
| 070H          | 1.000          | 0.228          |

TABLE 6. Anomaly detection rates across firmware groupings.

| Anomaly class | Reference class |
|---------------|----------------|
| 0309, 000F    | X              |
| 010G, 040H, 070H | 1.000         |

an analytical framework that paralleled that described in Section V-B, with the following two differences: 1) when comparing any two groups, since one was treated as the reference group and the other as the anomalous (test) group, each pair of groups yielded two (asymmetric) comparisons; and 2) because the reference group in each experiment was constructed using all of the available training data for that group, instead of using cross-validation, we used the proportion of test observations correctly labeled as anomalies as the performance metric.

Anomaly detection both by individual firmware and by major firmware grouping follow as expected from our pairwise classification results in Sections VI-A and VI-B. As shown in Table 5, detection rates at or near 100% are achieved when the firmware pair crosses the major version update (as discussed in the previous section) while the results are much poorer when the pair does not cross this major update boundary. Similarly, the anomaly detection rates by firmware grouping in Table 6 are high. These results suggest that, for example, a firmware anomaly detector can be successfully constructed provided the investigator is interested in detecting the presence of firmware versions that pre-date a certain version that represents a major milestone in the firmware update sequence (in our case, firmware version 010G).

E. COMPARING CLASSIFICATION PERFORMANCE ACROSS SSD OPERATING STATES

This section provides a comparison of classification rates using the four different identified operating states: read, write, idle, and off. The significant finding here is that classification can be accomplished using not only the active read and write states but also the low power idle state, as seen in Fig. 11. In this figure, classification rates based on data collected at idle are commensurate with classification rates in the two active states, read and write (at or near 100%). Unsurprisingly, classification rates are near chance when the SSD is in the off state and little to no current is flowing to it.

VII. CONCLUSION

In this paper, we characterized four SSD operating states and demonstrated the ability to distinguish between major firmware groupings within the same proprietary firmware family using current draw measurements in the low power idle state only. We achieved pairwise classification rates near 100% between firmware versions in different groupings which led to the ability to infer major updates in the firmware version timeline. We also developed an associated anomaly detector and demonstrated anomaly detection rates of 100% for firmware samples outside the reference grouping.

Follow-on work could potentially examine the accuracy of this approach across other manufacturer SSD firmware families or other memory technologies. The latter could include non-volatile novel memory technologies such as Intel’s phase change memory [39]. The use of other classification techniques as well as complex operations involving...
both reads and writes could also be explored to improve classification accuracy, particularly within the major firmware groupings.

Another natural follow-on would be an investigation of modified firmware. This could include manufacturer provided firmware versions that have been altered using the findings of [4] or malware versions found in the “wild.” The modification of open-source SSD firmware or the reverse-engineering of proprietary firmware also provide avenues for additional work as they could be used to further probe the nature of the detectable differences between various firmware versions.

APPENDIX

ADDITIONAL DETAILS OF FIRMWARE SPECIFICATIONS

As noted in Section III and summarized in Fig. 1, seven firmware updates have been released for the Crucial m4 SSD. While the work presented here focused on the latter five updates, all seven are discussed in detail in this appendix.

The first update to the baseline firmware (0001) included with the Crucial m4 SSD was 0002 released on 8 June 2011 [23]. This update provided additional margin for electromagnetic interference regulatory tests and performance improvements with Link Power Management (LPM) enabled. The latter was designed to alleviate pauses and hesitations that were being experienced with some host systems. While this update was optional, it was highly recommended for users experiencing performance issues with LPM enabled.

The second firmware update was 0009 on 25 August 2011 [23]. It provided a number of improvements including (directly from [23]):

- Improved throughput performance,
- Increase in PCMark Vantage benchmark score, resulting in improved user experience in most operating systems,
- Improved write latency for better performance under heavy write workloads,
- Faster boot up times,
- Improved compatibility with latest chipsets,
- Compensation for SATA speed negotiation issues between some SATA-II chipsets and the SATA-III device, and
- Improvement for intermittent failures in cold boot up related to some specific host systems.

This update received substantial news coverage as it increased the read speed of the drive by as much as 20% and introduced marginal improvements in write speed as well [40], [41].

Firmware update 0309 [24], [25] is the first of the five updates examined in this work. It was released on 13 January 2012 and fixed a critical drive error that caused a Blue Screen of Death (BSOD) after 5,184 hours of “on time” [34]. The reoccurring failure (every hour after initial occurrence) was due to an incorrect response to a SMART counter. While the update was strongly recommended for most users, it was incompatible with SAS expanders. The failure did not compromise user data on the drive.

Compatibility with SAS expanders and RAID cards was subsequently improved with update 000F [26], [27] which was released in April 2012. Modest performance gains were realized by improving throughput stability in heavy load scenarios and the update also improved data protection in the event of a power loss.

Firmware update 010G [28], [29] represented an important milestone as it was the first firmware update to include modifications specifically targeting Windows 8 platforms. These Windows 8 improvements reduced trim time by more than 50% and some users reported seeing a reduction in file delete times [42]. Other Windows 8 changes reduced power-on times from cold start and resume times from low power (sleep) modes. In addition to these Windows 8 enhancements, the update also included power consumption improvements for some notebook computers. It is worth noting that a number of users reported that the host system was no longer able to recognize the SSD following the update [43].

The next update, firmware version 040H [30], [31], was released in December 2012 and improved wear leveling algorithms. It also provided better recovery from unexpected power losses, corrected reporting errors in the SMART Drive Self Test, and improved the firmware update process for Windows 8.

Finally, the current firmware version for the Crucial m4 is 070H [32], [33], released in April 2013. Recommended for anyone using earlier firmware releases, it resolved a timing issue that could result in the host system hanging during power-up or resuming from sleep. At the time of release, Crucial considered this precautionary as no related failures had been reported by users.

REFERENCES

[1] L. Mearian, “Intel confirms data corruption bug in new SSDs, halts shipments,” Computerworld, Aug. 2009. [Online]. Available: https://www.computerworld.com/article/2525670/intelconfirms-data-corruption-bug-in-new-ssds-halts-shipments.html
[2] A. P. Barroso, V. H. Machado, and V. Cruz Machado, “Identifying vulnerabilities in the supply chain,” in Proc. IEEE Conf. Ind. Eng. Eng. Manage., Dec. 2009, pp. 1444–1448.
[3] L. Tokar, “Kingfast unknowingly sends counterfeit SSD with fake memory for review,” SSD Res., Feb. 2013. [Online]. Available: http://www.thessdreview.com/daily-news/latest-buzz/manufacturershipped-review-sample-ssd-contains-fake-stand-fake-memory/
[4] C. Meijer and B. Van Gastei, “Self-encryption deception: Weaknesses in the encryption of solid state drives,” in IEEE Proc. Secur. Privacy (SP), May 2019, pp. 72–87.
[5] J. E. Dunn. (Jul. 2019). Intel patches SSD firmware and microprocessor diagnostic tool. Naked Security by SOPHOS. [Online]. Available: https://www.nakedsecurity.sophos.com/2019/07/11/intel-patches-ssd-firmware-and-microprocessor-diagnostic-tool/
[6] J. Shey, J. A. Blanco, O. Walker, T. W. Tedesco, H. T. Ngo, R. Rakvic, and K. D. Fairbanks, “Monitoring device current to characterize trim operations of solid-state drives,” IEEE Trans. Inf. Forensics Security, vol. 14, no. 5, pp. 1296–1306, May 2019.
[7] D. Brown, O. Walker, R. Rakvic, R. W. Ives, H. Ngo, J. Shey, and J. Blanco, “Towards detection of modified firmware on solid state drives via side channel analysis,” in Proc. Int. Symp. Memory Syst. (MEMSYS), New York, NY, USA, 2018, pp. 315–320. doi: 10.1145/3240302.3285860.
[8] J. Melton, R. Rakvic, J. Shey, H. Ngo, O. Walker, J. Blanco, D. Brown, L. Medcalf, and K. D. Fairbanks, “Inferring File System of Solid State Drives based on Current Consumption,” in Proc. IEEE 7th Annu. Int. Conf. CYBER Technol. Autom., Control, Intell. Syst. (CYBER), Jul. 2017, pp. 72–76.
JUSTIN A. BLANCO (Senior Member, IEEE) received the Sc.B. degree in neuroscience from Brown University, in 1999, the M.S. degree in mechanical engineering from Stanford University, in 2003, and the Ph.D. degree in bioengineering from the University of Pennsylvania, in 2010. He was a Postdoctoral Research Associate at the Translational Neuroengineering Laboratory, University of Pennsylvania, from 2010 to 2011, before joining the faculty at the United States Naval Academy, in 2011. Prior to his doctoral studies, he has served on the faculty at Deerfield Academy, Deerfield, MA, USA, and was a Design Engineer at Medtronic. He is currently an Associate Professor with the Department of Electrical and Computer Engineering, United States Naval Academy, where he heads the Signal Processing and Data Science Research Group. His areas of research interests include biomedical signal analysis and statistical learning theory and applications in neurological diseases, implantable medical devices, and digital security.

RYAN RAKVIC received the B.S. degree in computer engineering from the University of Michigan, and the M.S. and Ph.D. degrees with Carnegie Mellon University. He is currently in his 14th year as a Professor with the Electrical and Computer Engineering Department, United States Naval Academy, Annapolis, MD, USA. Prior to that, he spent five years at the Microarchitecture Research Lab, Intel Corporation, Santa Clara, CA, USA. During his time at Intel, he invented creative ways to increase the performance of microprocessors. At the Academy, he has spent time exploring other areas of computer architecture, including studying Field Programmable Gate Arrays (FPGAs), Graphics Processing Engines (GPUs), and embedded smartphone designs in the context of not just performance but also energy efficiency. Most recently, he has branched out to the computer security field, where he is currently working at the crossroads of side-channel analysis and machine learning.

ANN VANLEER received the B.S. degree in weapons and controls systems engineering from the U.S. Naval Academy, in 1996, the M.S. degree in electrical engineering from the University of Washington, in 1997, and the Ph.D. degree in bioengineering from the University of Pennsylvania, in 2013. She was a Postdoctoral Research Associate at the Translational Neuroengineering Laboratory, University of Pennsylvania, from 2013 to 2014, before joining the faculty at the United States Naval Academy, in 2014. She has served as the Naval Surface Warfare Officer. Prior to her doctoral studies, she has served on the faculty at The Key School, Annapolis, MD, USA. She is currently an Assistant Professor with the Department of Electrical and Computer Engineering, United States Naval Academy. Her areas of research interests include biomedical signal analysis and machine learning.

DANE BROWN received the B.S. degree in electrical engineering from the U.S. Naval Academy, in 2005, the M.S. degree in electrical engineering from the Naval Postgraduate School, Monterey, CA, USA, in 2006, and the Ph.D. degree in computer engineering from the Virginia Polytechnic Institute and State University, in 2019. He began his career as a Navy Submarine Warfare Officer after graduating from the Navy Nuclear Power School and the Navy Nuclear Power Training Command in Goose Creek, SC, USA. He is currently an Assistant Professor with the Cyber Science Department, U.S. Naval Academy. His research interests include reverse engineering, vulnerability discovery, privacy and trust, instruction set architectures, communications protocols, digital forensics, and education in cyber security and STEM fields.

JAMES SHEY received the B.S. degree in electrical engineering from the U.S. Naval Academy, in 2003, the M.S. degree in electrical engineering from the University of Maryland, College Park, MA, USA, in 2008, and the MEM from Old Dominion University, in 2008. He is currently pursuing the Ph.D. degree with the University of Maryland, Baltimore County, MA. He was an Instructor at the U.S. Naval Academy, from 2008 to 2010, and resumed these duties, in 2015. Between these time periods, he has served in various assignments, both ashore and afloat as a Submarine Warfare Officer, trained in naval nuclear propulsion. His research interests include integrated circuit aging effects, parallel programming, GPU acceleration, SSD security, and machine learning.

GREGORY L. SINSLEY received the B.S. degree in electrical engineering from the Grove City College, in 2005, and the Ph.D. degree in electrical engineering from the Pennsylvania State University, in 2012. He was an Electrical Engineer with Applied Technology, Inc., from 2012 to 2016, and an Electronics Engineer with the United States Naval Research Laboratory, from 2016 to 2018. He has been an Assistant Research Professor with the Electrical and Computer Engineering Department, United States Naval Academy, since 2018. His research interests are in optimization and control of power systems, multisensor data fusion, anomaly detection in cyber-physical systems, and side-channel analysis.

HAU T. NGO received the B.S. and M.S. degrees in computer engineering and the Ph.D. degree in electrical and computer engineering from Old Dominion University, in 2001, 2003, and 2007, respectively. He is currently an Associate Professor with the Electrical and Computer Engineering Department, U.S. Naval Academy, Annapolis, MD, USA. His research interests include biometric signal processing, power-delay-area efficient design methodologies for image and video processing systems, real-time signal processing on field-programmable gate arrays (FPGAs), hardware-software co-design of high-performance embedded systems, and digital forensics.

ROBERT W. IVES (Senior Member, IEEE) received the B.S. degree in mathematics from the United States Naval Academy, the M.S. degree in electrical engineering from the U.S. Naval Postgraduate School, and the Ph.D. degree in electrical engineering from the University of New Mexico. He served 20 years and has retired from the Navy as a Nuclear Submarine Officer. One of his tours in the Navy was as a Military Research Associate at Sandia National Laboratories, Albuquerque, NM, USA, where he was engaged in pattern recognition and image compression research. He is currently in his 17th year as a Professor with the Electrical and Computer Engineering Department, United States Naval Academy, Annapolis, MD, USA. His research interests include image and video processing, pattern recognition, and machine vision.