Secure Deletion of Data from SSD

Akli Fundo[1], Aitenka Hysi[2] Igli Tafa [3]
Polytechnic University of Tirana,
Computer Engineering Department

***Abstract*—The deletion of data from storage is an important component on data security. The deletion of entire disc or special files is well-known on hard drives, but this is quite different on SSDs, because they have a different architecture inside, and the main problem is if they serve the same methods like hard drives for data deletion or erasing. The *built-in* operations are used to do this on SSDs. The purpose of this review is to analyses some methods which are proposed to erase data form SSDs and their results too, to see which of them offers the best choice. In general we will see that the techniques of erasing data from entire disc from hard drives can be used also on SSDs, but there's a problem with bugs. On the other hand, we cannot use the same techniques of erasing a file from hard drives and SSDs. To make this possible, there are required changes in FTL layer, which is responsible for mapping between logic addresses and physical addresses.**

Keywords—deletion-data; SSD; FTL layer; logic address; physical address

I. INTRODUCTION

Nowadays, corporations and agency's store their data's in digital media, managing them is becoming important. In this article we will focus on challenges of SSDs for erasing the information, and some suggestions from different researches and experimental results. Data erasing is an important process and different techniques are include like built-in in ATA or SCSI commands. This techniques are effective on HDDs, where we can store the entire disc or specific files, but it's not the same thing with SSDs, because SSDs and HDDs have a different technology and algorithms of managing the information. SSDs have an indirect layer between the logic address that computer systems use to access data and the address that identify the physical storage [8]. The differences between SSDs and hard drives make it unclear of which techniques or commands are worthy on both of them. An experiment of [8] make this clear: they have write a structured data model to the drive, apply the deletion techniques, dismantle the drive and extract the data directly from the flash chips using a flash testing system [8]. To delete one file they made changes on FTL layer [1]. From the articles I've read, I have seen that there are solutions about this problem, but the performance is decreasing. On the other section I would like to show some of the deletion problems and what different engineers have done to solve them. But firstly I would like to show from [9] four levels of clearing(sanitizing) that a storage media could have:

The first level is *logic clearing*. The deleted data on logic way cannot be salvage through standard interface hard-ware like ATA or SCSI commands. The user can delete logically one file or the entire disc by overwriting respectively the whole disc or a part of it.

The second level is *digital clearing*. In this case it's impossible to recuperate the data in a digital way.

The third level is *analog clearing*. This level can damage the signal which encodes the data, and it's impossible to rebuilt the signal even with very sensitive equipments. An alternative way to "hide" the bits is *cryptographic clearing* [4]. In this level the media uses a key for encryption and decryption of the data which enter and exit. But this is not a secure way, because someone can extract the key from physical media and can avoid the encryption.

II. RELATED WORKS

SSDs have specific characteristics. The traditional magnetic discs are composed by sectors 4 KB. The sector is the smallest unit of data which can be read or written on the disc. SSDs operate in another way, after the minimal number of bytes which can be read vary from those which can be written or delete. The flash banks have a typical block size, from 16 KB to 512 KB and the minimal number of bytes written simultaneously is 4 KB [12]. Otherwise from magnetic discs, the flash media firstly must delete a whole block before writing new data. SSDs don't have mechanical part so they don't have rotation or searching latency [11]. That's why they have a good performance. But there's a problem with the "*wear*" phenomenon, which is present over times. Every block of SSD can be deleted in finite times and after that it can not be written anymore. To fix this problem engineers have applied "wear leveling" in I/O controller. This is a group of algorithms used by controller to make a same allocation (diffusion) of deleting cycles on every block of the memory flash. In this way, no one of the blocks cannot be deleted not normally, to avoid SSDS fail [12]. Another way to to fix "*wear*" problem and the deleting before writing engineers have proposed a hybrid architecture SSD-HDD called HPDA [13], which is more reliable and increases the performance. From [3] SSD uses a flash memory to store the data. This memory is divide into pages and blocks. The program operations interfere in this pages and change "1" to "0". The clearing operations are applied in the blocks and set all the bits of the blocks in "1". Usually there are 64-256 pages/blocks. FTL manages the mapping between logic blocks addresses (LBA), we can see them from ATA or SCSI interfaces and physic pages of flash memory. There is a mismatch of this two operations, so it's not possible a directly update of the LBA sector. Instead of modifying the sector, FTL will write the new content of sector in another place and will update the map.

The flash memory will keep the old version of data in a digital form[8]. Since directly updates are not possible as I said upper, the overwriting techniques which work on hard drives may not work on SSDs. This techniques suppose that overwriting of a part in a LBA area will result in overwriting on the same physic place. Except from FTL, the engineers have proposed **CAFTL** [14] (Content-Aware Flash Translation Layer) to reduce in an effective way the traffic writing on memory flash, removing the unnecessary duplicated writing. They have realized this by join together the parity data, which increase the efficiency of garbage collection and "wear leveling".

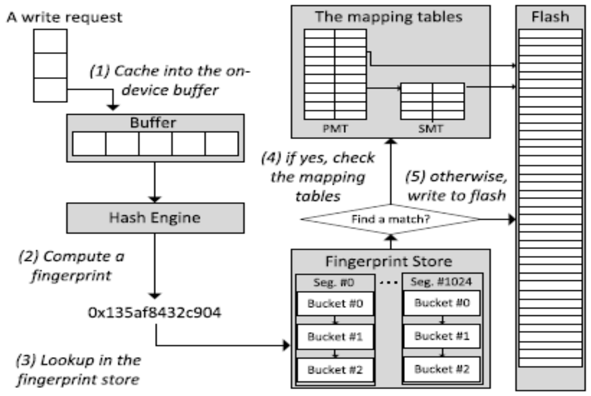

A write request
(1) Cache into the on-device buffer
Buffer
Hash Engine
(2) Compute a fingerprint
0x135af8432c904
(3) Lookup in the fingerprint store
The mapping tables
PMT
SMT
(4) if yes, check the mapping tables
(5) otherwise, write to flash
Find a match?
Fingerprint Store
Seg. #0
Seg. #1024
Bucket #0
Bucket #1
Bucket #2
Flash

Fig. 1. CAFTL architecture [14]

The figure 1 shows the architecture of a CAFTL. It eliminates the duplications of writing and redundant data through a combination of the non duplications: in-line and out-of-line.

It examines the entry data and delete the redundant data before the writing process in flash. Anyway, it doesn't guarantee that all duplicated writes will delete immediately, that's why CAFTL scans flash memory and reduce the redundant data out-of-line.

Figures 2 and 3 shows exactly the improvement on eliminating duplicate writes with 24,2 % and the extended flash memory with 31,2 %. Here offline means that duplications are eliminated offline, no-sampling refers to the purely CAFTL with a sampling unit of 128 KB [14].

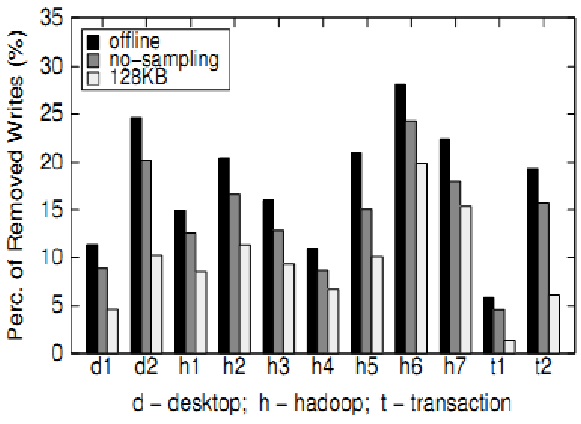


Fig. 2. Percentage of removed duplicate writes.[14]

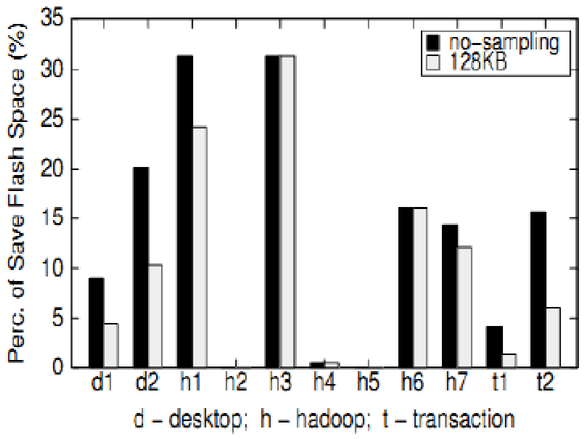


Fig. 3. Percentage of extended flash space.[14]

After these experiments engineers have seen that offline is the optimal case, but purely CAFTL is able to eliminate the duplicate writes in a significative way. Also the analog clearing is more complex on SSDs. [4] examines the garbage problem of data in flash, DRAM, SRAM and EEPROM and the "cold boot" attacks.

The simplest method is that the voltage level in the floating gate of an erase flash cell may vary, depending on the value it has before the erase command. The quantity of digital garbage may be very big. The SSDs which are tested contain 6-25 % more physical storage than they show on their logic capacity[8]. Figure 4 shows the garbage on the SSDs. There are created 1000 small files on the SSDs, the driver is dismantled and analyses. For some of this files SSD contains up to 16 old copies. This copies were created during garbage collection and un directly updates.

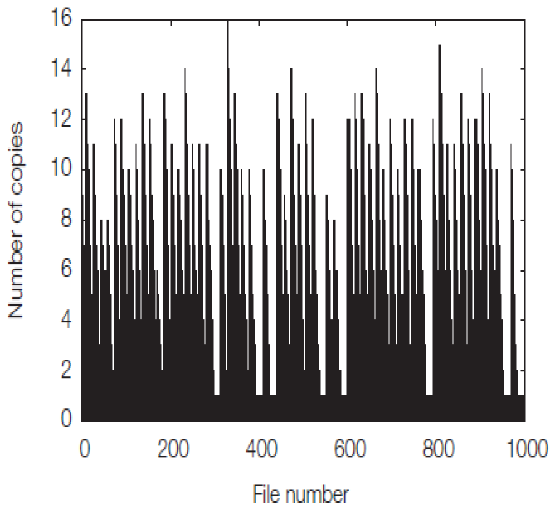


Fig. 4. Multiple copies creating from FTL, duplicating files up to 16 times.[8]

The differences between SSDs and hard drives described in upper sections will create a disconnect between what a user expects and how drive behaves. Someone who has an SSD may use a clearing technique which is characteristic for hard drive, in way to make the data inaccessible, but the data will stay on the drive and they can be extracted with sophisticated methods.

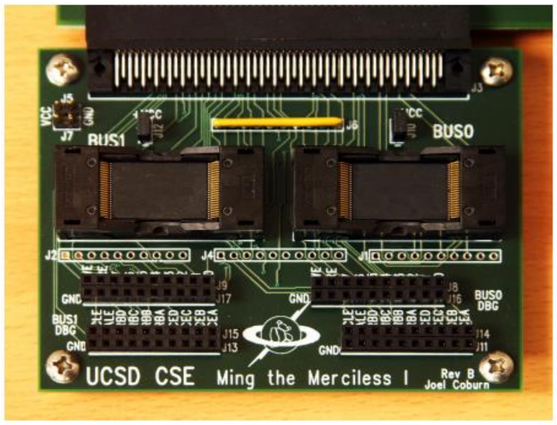


Fig. 5. FPGA based on flash testing hardware.[7][8]

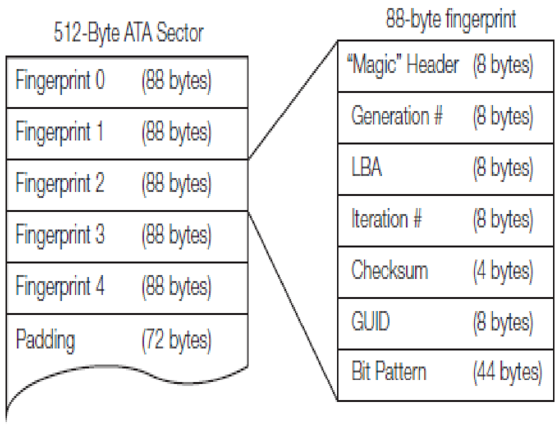
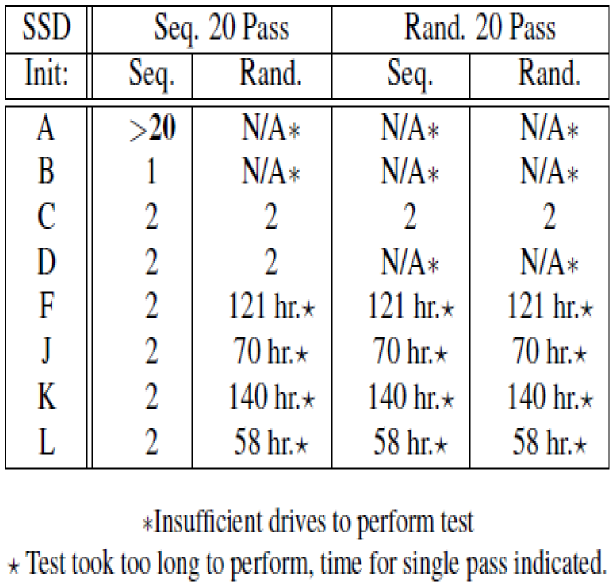


Fig. 6. Fingerprint structure. The easily-identified fingerprint simplifies the task of identifying and reconstructing remnant data.[8]

The engineers of [8] had verified the second level which is digital clearing by using the lowest level of digital interface: the pins of individual flash chips. To verify this operation, they wrote an identifiable data model called *fingerprints* and then they applied the clearing techniques under test. The fingerprint makes possible the identification of digital garbage on the chips. It also includes a sequence number that is unique at entire fingerprints. The figure 6 shows the fingerprint structure. According to the fig. 6 every fingerprint is 88 byte long repeats five times in a 512 byte ATA sector. Another method described in the article is overwriting every logic block address on the drive. This is the main method for many disc deletions. The different bits aim to change a lot of physical bits as possible on the drive, make it harder to recover the data with analog ways.

TABLE I. THE SOFTWARE OF OVERWRITING WHOLE DISC. THE NUMBER OF EACH COLUMN SHOWS THE NUMBER OF THE STEPS FOR DELETING THE DATA FROM THE DRIVE.[8]



SSD	Seq. 20 Pass		Rand. 20 Pass	
Init:	Seq.	Rand.	Seq.	Rand.
A	>20	N/A∗	N/A∗	N/A∗
B	1	N/A∗	N/A∗	N/A∗
C	2	2	2	2
D	2	2	N/A∗	N/A∗
F	2	121 hr.⋆	121 hr.⋆	121 hr.⋆
J	2	70 hr.⋆	70 hr.⋆	70 hr.⋆
K	2	140 hr.⋆	140 hr.⋆	140 hr.⋆
L	2	58 hr.⋆	58 hr.⋆	58 hr.⋆

∗Insufficient drives to perform test
⋆ Test took too long to perform, time for single pass indicated.

As it looked, these bits are important for SSDs. According to the experiments that the engineers have made, they have seen that some SSDs compress their data before storing them, they will write fewer data on flash. They suggest that for maximum effectiveness, SSD overwriting procedures should use random data. One of the drivers tested from them showed that it used compression and encrypted the data, but they can't verify the erasing process.[8]

They have tested eight drivers which don't use encryption and table 1 shows the results of these tests. The numbers show how many generations of data were needed to delete the drive. For some drives, they realized that random writes were to slow so they didn't do the test to these drives. In some cases they overwrite the drive twice which was enough to erase the disc, no matter how was its previous state. But they found exceptions on drive A, because some of the data weren't erased totally.

They compared this technique with the first one and find out that this is more reliable, but it's not totally secure, because of the type of the drivers they use.

Another method that [8] evaluate for erasing data was degaussing. Degaussing is the eliminating process or reducing the undesired magnetic fields [10]. Degaussing is fast and effective because it removes the low level formatting and damages the drive motor. They don't except this method will work including the architecture of SSDs. And the experiment made in [6] shows that after degaussing, nothing happened to the data, so this method fails on SSDs.

III. Conclusion

This article was focused on the proposed techniques from different engineers to sanitize data from SSDs. These techniques were based on hard drives, because they were successfully in there. From the results, the first method had a half success (50:50), the second method was more successfully except some fails it had and the third method was worthless, because it doesn't guarantee anything. So the problem of sanitizing data from SSDs actually is very present nowadays and engineers are trying to find ways to do this more reliable. From [6] an efficient way is to combine the techniques to each other for a safe and verifiable sanitizing data from SSD.

References

[1] J. Lee, J. Heo, Y. Cho, J. Hong, and S. Y. Shin. Secure deletion for nand flash file system. In SAC '08: Proceedings of the 2008 ACM symposium on Applied computing, pages 1710–1714, New York, NY, USA, 2008. ACM.

[2] A. Birrell, M. Isard, C. Thacker, and T. Wobber. A design for high-performance flash disks. Technical Report MSR-TR-2005-176, Microsoft Research, December 2005.

[3] E. Gal, S. Toledo. Algorithms and data structures for flash memories. ACM Comput. Surv., 37(2):138–163, 2005.

[4] P. Gutmann. Secure deletion of data from magnetic and solid-state memory. In SSYM'96: Proceedings of the 6th conference on USENIX Security Symposium, Focusing on Applications of Cryptography, pages 8–8, Berkeley, CA, USA, 1996. USENIX Association.https://www.cs.auckland.ac.nz/~pgut001/pubs/secure_del.html

[5] USB Implementers Forum. Universal Serial Bus Mass Storage Class Specification Overview, September 2008.

[6] S. Swanson and M.Wei. Safe: Fast, verifiable, sanitization for SSDs. http://nvsl.ucsd.edu/sanitize/, October 2010.

[7] L. M. Grupp, A. M. Caulfield, J. Coburn, S. Swanson, E. Yaakobi, P. H. Siegel, and J. K.Wolf. Characterizing flash memory: Anomalies, observations and applications. In MICRO'09: Proceedings of ..., New York, NY, USA, 2009. ACM, IEEE.

[8] M Wei, L.Grupp, F.Spada, and S.Swanson. Reliably Erasing Data From Flash-Based Solid State Drives. In FAST' 2011.

[9] Royal Canadian Mounted Police. G2-003, Hard Drive Secure Information Removal and Destruction Guidelines.

[10] http://en.wikipedia.org/wiki/Degaussing

[11] "Solid-state drive",2012 http://en.wikipedia.org/wiki/Solid-state_drive.

[12] David Bartizal,Thomas Northfield,"Solid State Drive Performance", www.csee.umbc.edu/~squire/images/**ssd**2.pdf

[13] Bo Mao,Hong Jiang,"HPDA:A hybrid parity-based disk array for enhaced performance and reliability" , Parallel & Distributed Processing , 2010 IEEE International Symposium

[14] F.Chen,T.Luo,"CAFTL: A content-aware flash translation layer enhancing the lifespan of flash based solid state drives",9th USENIX Conference on file and storage technologies, 2011, www.cse.ohio-**state**.edu /~fchen/paper/papers/fast11.pdf