WAS-Deletion: Workload-Aware Secure Deletion Scheme for Solid-State Drives

Bingzhe Li*, and David Du‡
*Oklahoma State University, Stillwater, OK, USA
‡University of Minnesota – Twin Cities, Minneapolis, MN, USA
bingzhe.li@okstate.edu, du@umn.edu

2021 IEEE 39th International Conference on Computer Design (ICCD) | 978-1-6654-3219-1/21/$31.00 ©2021 IEEE | DOI: 10.1109/ICCD53106.2021.00047

***Abstract*—Due to the intrinsic properties of Solid-State Drives (SSDs), invalid data remain in SSDs before erased by a garbage collection process, which increases the risk of being attacked by adversaries. Previous studies use erase and cryptography based schemes to purposely delete target data but face extremely large overhead. In this paper, we propose a Workload-Aware Secure Deletion scheme, called WAS-Deletion, to reduce the overhead of secure deletion by three major components. First, the WAS-Deletion scheme efficiently splits invalid and valid data into different blocks based on workload characteristics. Second, the WAS-Deletion scheme uses a new encryption allocation scheme, making the encryption follow the same direction as the write on multiple blocks and vertically encrypts pages with the same key in one block. Finally, a new adaptive scheduling scheme can dynamically change the configurations of different regions to further reduce secure deletion overhead based on the current workload. The experimental results indicate that the newly proposed WAS-Deletion scheme can reduce the secure deletion cost by about 1.2x to 12.9x compared to previous studies.**

I. Introduction

In the era of big data, digital information must be stored in non-volatile storage media. Many technologies have been investigated in Solid State Drive (SSD) [1]–[3]. However, as increasing security concern of sensitive data, how to store data with privacy and security has become more and more critical [1], [4], [5].

Techniques for SSD secure deletion can be categorized into two groups including 1) delete target and invalid data with a command [6], [7], and 2) periodically delete all invalid data [1], [8], [9]. For the first method, when a secure deletion command is issued to delete target data, SSDs need to delete the target valid data and all their corresponding invalid data (old versions of the target data). However, this method faces a large overhead of tracking and deleting all those invalid pages associated with the target data in the past. The other method periodically deletes all the invalid data or invalid encryption keys [1], [8], [9] to protect the invalid pages from unveiling to attackers. When a secure deletion is issued for the target data, SSDs first invalidate the target data and then securely delete all invalid pages to guarantee 100% protection for the target data and all existing invalid data. In this paper, we mainly focus on the periodical secure deletion scheme to reduce the secure deletion overhead.

To periodically delete invalid data in SSDs, some previous studies [1], [8]–[10] used erase the blocks and delete encryption keys to make target data inaccessible. More details of those schemes are discussed in Section II. The disadvantages of all those existing studies is that they did not consider the effect of workload access patterns. As a result, the mixture of invalid and valid pages in the same blocks (a block is a unit of erasure in SSD) or in the same chunk during the secure deletion may induce a large overhead of migration and erase.

In this paper, a new Workload Aware Secure Deletion scheme called WAS-Deletion is proposed to reduce the overhead of secure deletion for SSDs. The WAS-Deletion scheme efficiently splits invalid and valid pages into different blocks or chunks according to their historically accumulated update frequencies and update request sizes. Consequently, the pages likely to be in the same states (either invalid or valid) are mostly located in the same chunks, mitigating the overhead of migration and erase during secure deletion. Moreover, we propose a vertically encrypted allocation that follows the same write direction of pages in blocks. As a result, this vertically encrypted allocation can further reduce the fragments of invalid and valid pages in one chunk. Finally, several regions of different configurations of data chunks are used. The chunks in different regions have different sizes and each region is applied with an individual deletion scheme. An adaptive scheme is also applied by the WAS-Deletion scheme to dynamically schedule incoming requests in different regions based on their access patterns. This adaptation further reduces the overhead of secure deletion compared with previous studies.

The structure of the paper is as follows. Section II describes the background of SSD and related work of secure deletion. The proposed WAS-Deletion scheme is introduced in Section III. Section IV shows the experimental results compared to previous studies. Finally, some conclusions are presented in Section V.

II. Background and Related Work

Secure deletion in SSDs is responsible for deleting both target data pages and their associated invalid pages to protect data privacy and security. The periodic secure deletion is to periodically delete all invalid data [1], [8], [9] to make identifying or searching associated invalid pages of target data easier when a secure deletion is issued. Compared to direct deletion, the advantage of the periodical secure deletion is to provide a mechanism to protect all invalid data on SSDs periodically. Moreover, the periodical deletion can delete target

978-1-6654-3219-1/21/$31.00 ©2021 IEEE
DOI 10.1109/ICCD53106.2021.00047

files and then their associated invalid data accumulated in the current period.

To periodically delete invalid pages in SSDs, two basic schemes, erase-based and cryptography-based schemes, were proposed. For the erase-based scheme, the idea is to periodically erase all invalid data for secure deletion. In the cryptography-based scheme [10], data pages are encrypted with security keys before being written to the flash memory. Meanwhile, the keys used in the encryption are also stored in flash memory. During secure deletion, the SSD only needs to delete/erase the keys of invalid data pages. To reduce the key space, pages across multiple blocks can share the same key. These multiple chunks comprise one **chunk**. The number of blocks in a chunk is called **chunk size** (C_z). Liu *et al.* [1] proposed ErasuCrypto, which combines both erase-based and cryptography-based schemes to find a minimum secure deletion cost (as seen in Eq. (1)) by applying either cryptography-based or erase-based scheme.

$$cost = \#Mgr + k \times \#Era \tag{1}$$

where $\#Mgr$ and $\#Era$ indicate the numbers of migrations and erases, respectively for secure deletion. k is the coefficient to indicate the ratio between erase cost and migration cost.

Overall, previous studies [1], [9], [11] passively address the periodical secure deletion issue with little intent to optimize data scheduling and management. Therefore, there is an opportunity to further reduce the overhead of secure deletion by scheduling and managing incoming data according to the access patterns of workloads.

III. Adaptive WAS-Deletion Scheme

We propose an adaptive WAS-Deletion scheme involving the optimization of those factors. There are two major steps in the WAS-Deletion scheme. The first step is the classification process based on historical update frequencies and write request sizes to classify data into different categories (regions). The second step is to apply different deletion schemes to different regions.

Different regions use different chunk sizes. To satisfy the constraint of the capacity number of key blocks, each region's number needs to follow Eq. (2).

$$\begin{aligned} \sum_{i=0}^{N-1} B_i &\leq B_{max} \\ \sum_{i=0}^{N-1} B_i * C_{zi} &= Cap \end{aligned} \tag{2}$$

where N is the number of regions. B_i indicates the number of key blocks in the i^{th} region. C_{zi} is the chunk size for the i^{th} region. Cap is the total number of data blocks in a SSD. B_{max} is the maximum number of key blocks limited by the SSD capacity. In this paper, $Cap = 250GB$ and $B_{max} = 250$ by default. The ErasuCrypto scheme is one special case under Eq. (2) with N=1, $B_0 = 250$ and $C_{z0} = 8$.

In the first step of the WAS-Deletion scheme, the SSD keeps accumulating historical access pattern information and the default unit size is block size (512KB). First, the numbers of updates and write request sizes for each block are recorded in **Freq_TBL** and **Size_TBL**. Supposing we use N regions, N centroids are computed as the representative values for each region category. Once obtaining the centroids, a new coming request in the next period can be classified based on those centroids. The Euclidean distance is computed between the block number (**S**) of the request and each centroid value. Then, the request should be assigned to a temporary region of which centroid and its S value achieve the minimum Euclidean distance.

After that, the second layer classification will re-classify the request based on the write request sizes. Using the vertical encryption allocation, a large request size can reduce the migration overhead because the large-size updates will invalidate several consecutive pages in one chunk encrypted with the same key. So, in our algorithm, if the average write request size is two times larger than the chunk size, the temporary region classified by the first layer classification will be shifted to one left (i.e., the region with a larger chunk size). In contrast, if the average request size is too small (such as smaller than two pages), the region of the block will be degraded to the region with a smaller chunk size. Finally, to satisfy Eq. (2) an inspection function is used to check the current state of regions. If the current state of regions violates Eq. (2), the inspector starts from the largest region (Region#N-1) to adjust regions to a nearby region (Region#N-2) until the capacities of all regions satisfy Eq. (2).

Algorithm 1 WAS-Deletion

```
1: procedure SECURE DELETION
2:     if Region#0 then
3:         Erase-based deletion for the region#0
4:     else if Region#N-1 then
5:         Cryptography-based deletion for the region#N-1
6:     else
7:         for each block do
8:             Compute cost_erase by searching the whole block
9:             Compute cost_cry by searching multiple rows in the block
10:            if cost_erase < cost_cry then
11:                Cryptography-based deletion for this chunk
12:            else
13:                Erase-based deletion for this chunk
```

In the second step, the WAS-Deletion scheme as shown in Algorithm 1 applies different secure deletion schemes to different regions. Moreover, different regions have different chunk sizes to limit the key block overhead. In WAS-Deletion, the blocks in the coldest region (access frequencies of requests are low) contain the least number of updates. Thus, we can use the erase-based scheme with little migration overhead for the coldest region (Lines 2-3 of Algorithm 1) and no encryption is needed in this region. There are two advantages. The first one is that without encryption and decryption processes data access latency will be reduced. The other is that no key block is needed and thus it saves key block spaces for other regions. In contrast, the hottest region (Region#N-1) contains the data with the most frequent updates. Although the data are highly frequently updated in this region, these data may not keep the

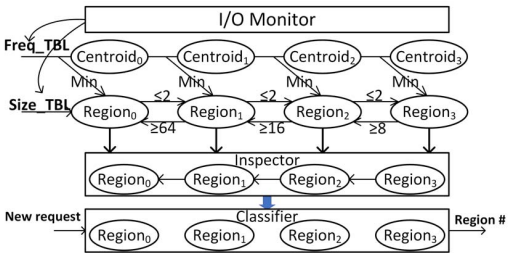

Fig. 1: A data-flow of the WAS-Deletion classification.

TABLE I: SSD configuration.

Parameter	Value	Parameter	Value
SSD capacity	250GB	Read latency	20us
Page Size	4KB	Program Latency	200us
Block size	512KB	Erase latency	1.5ms
# of pages/block	128	# of key blocks	250

same pace to be updated due to the small number of categories. Thus, the cryptography-based scheme is used in the hottest region (Lines 4-5). Since the number of invalid data pages varies in the hottest region, the smaller numbers (chunk size) of pages that share the same key will help reduce the migration and erase overheads. Thus, we set the chunk size of 1 for this region, which means that one encryption key only encrypts one page. For the other regions with the middle level of update frequencies, their data are hotter than the coldest region and have data colder than the hottest region. Therefore, since the direction of encryption and write is the same, we can search each block to compute the overhead of erase and crypto based schemes (Lines 8-9) based on Eq. (1). Then, the corresponding scheme will be used to minimize the secure deletion overhead (Lines 10 - 13). The chunk sizes of those regions are set to 4 and 8, respectively.

Figure 1 indicates an example of WAS-Deletion dynamic classification with four regions. First, the I/O monitor collects the I/O update frequencies and I/O request sizes for each block. Then, four centroids can be computed based on the accumulated I/O update frequencies. The minimum Euclidean distance determines in which region the corresponding block initially resides. After that, the algorithm needs to check the average request size of each block. If the request size is too small, the block's region should be shifted right by one. If the request size is too large, the region will be shifted left by one. Otherwise, the block keeps in the current region. The last step is that the inspector checks whether the current configuration of regions satisfies the requirement of Eq. (2). If not, some blocks will be re-arranged until the requirement of Eq. (2) is satisfied. Finally, a classifier is built. Once an update request is coming to SSD, we first compute the block number of the request. Then the classifier will tell which region the request should be scheduled to.

IV. Evaluation

In this section, we make comparisons between different secure deletion schemes.

TABLE II: MSR Cambridge trace configurations [12].

	Number of IOs (Millions)		Total request size (GB)	
	Write	Read	Write	Read
prn_0	4.98	0.602	45.96	13.12
prn_1	2.77	8.46	30.78	181.35
proj_1	2.50	21.14	25.58	750.36
usr_1	3.86	41.43	56.13	2079.23
usr_2	1.99	8.58	26.47	415.28
hm_0	2.58	1.42	20.47	9.96
rsrch_0	1.30	0.13	10.82	1.39
stg_1	0.80	1.40	5.99	79.52
prxy_0	12.14	0.38	53.80	3.05
ts_0	1.49	0.32	11.34	4.13

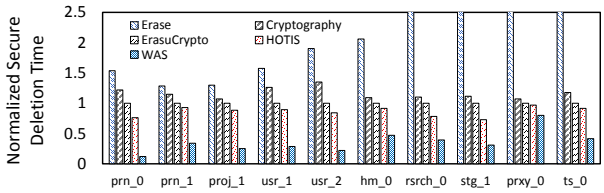

Fig. 2: Normalized secure time comparison between four schemes.

A. Evaluation Environment Setup

The SSD configuration is listed in Table I. The SSD capacity is 250GB. The number of key blocks is set to 250. It means that the chunk size is 8 for the ErasuCrypto scheme. The encryption algorithm uses the AES algorithm with 128 bits (AES-128). The real traces used are Cambridge MSR traces [12]. The experimental results are obtained based on the SSDsim simulator [13]. We assume that the secure deletion command is triggered after replaying each 168-hour trace.

Five schemes are compared: erase-based, cryptography-based, ErasuCrypto [1], HOTIS [2], and WAS-Delete (WAS) schemes. The erase-based, cryptography-based and ErasuCrypto as introduced in Section II are three most relevant secure deletion works. HOTIS is a classic hot/cold data classification scheme in SSDs. The WAS-Delete (WAS) scheme uses four regions and it adaptively changes the region configuration based on workload access patterns. Moreover, the total number of key blocks for ErasuCrypto, Cryptography-based, and WAS schemes is 250. Three metrics, normalized secure deletion execution time (**secure time**), **number of page migrations**, and **number of block erases**, are used to indicate the overhead during the secure deletion process.

B. Overall Performance Comparison

Figure 2, Figure 3, and Figure 4 indicate the normalized secure time, number of page migrations, and number of block erases between erase-based, cryptography-based, ErasuCrypto [1], WAS-Delete schemes, respectively. The secure time of the ErasuCrypto scheme is normalized to one. As seen in Figure 2, the erase-based scheme obtains the worse secure time and the cryptography-based scheme is the second worse secure deletion scheme. The proposed WAS scheme achieves the lowest secure time, which achieves about 3.8x - 12.9x, 1.3x - 10.1x, 1.2x - 8.3x, and 1.2x - 6.9x secure time

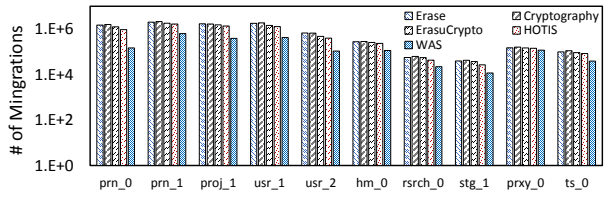

Fig. 3: Number of migrations between four schemes (y-axis is log-scale).

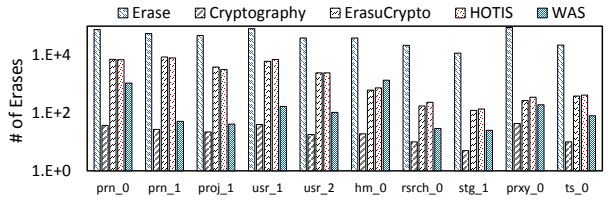

Fig. 4: Number of erases between four schemes (y-axis is log-scale).

reduction compared to erase, cryptography, ErasuCrypto, and HOTIS schemes, respectively. In summary, the proposed WAS-Deletion scheme obtains on average about 2.89x - 7.25x secure time reduction compared to the other four schemes among different workloads.

The details of secure deletion cost are provided based on page migrations and block erases. For the number of page migrations as seen in Figure 3, the WAS-Deletion scheme largely reduces the page migration overhead compared to the other four schemes. For the number of block erases as seen in Figure 4, the cryptography-based scheme achieves the least number of erases because there is no data block erase and the erases only happen on the key blocks. However, the data pages associated with the deleted keys become invalid and need to be garbage collected later. Though the proposed WAS-Deletion scheme achieves the second least number of erases, the total secure deletion time of the WAS-Deletion scheme is much smaller than the cryptography-based scheme.

There are three main reasons that the WAS-Deletion reduces both erase and migration overheads. First, the WAS-Deletion scheme distinguishes the data 'hotness' and separates them into different regions. The second reason is that the write and encryption follow the same direction and the large I/O write requests can invalidate several pages in one chunk at the same. As a result, during the secure deletion, the number of valid pages to be migrated is reduced and so the migration overhead is decreased. Finally, the adaptive scheme selects more efficient region configurations and achieves more accurate classification so that the data are efficiently separated to obtain less migration and erase overheads.

V. Conclusion

In this paper, a Workload-Aware Secure deletion scheme called WAS-Deletion is proposed. First, the WAS-Deletion scheme efficiently splits invalid and valid pages into different blocks or chunks according to the historical accumulated update frequency and update request size. Second, a vertical encryption scheme is applied to the scheme which reduces the migration overhead. Third, the request size is used as a 'hotness' factor to cluster similar data in the same region associated with the write request size. Finally, an adaptive scheme is used to adjust the region configuration according to the workload access patterns. In the experimental results, the newly proposed WAS-Deletion scheme is capable of reducing the secure deletion time about 1.2x to 12.9x compared to previous studies. Moreover, the breakdown analysis and investigation about different design parameters are provided and contribute to a deeper understanding of the trade-offs in the secure deletion of SSDs.

VI. Acknowledgement

This work was partially supported by NSF I/UCRC Center Research in Intelligent Storage and the following NSF awards 1439622, and 1812537.

References

[1] C. Liu, H. A. Khouzani, and C. Yang, "Erasucrypto: A light-weight secure data deletion scheme for solid state drives," *Proceedings on Privacy Enhancing Technologies*, vol. 2017, no. 1, pp. 132–148, 2017.

[2] J. Gu, C. Wu, and J. Li, "Hotis: A hot data identification scheme to optimize garbage collection of ssds," in *2017 IEEE International Symposium on Parallel and Distributed Processing with Applications and 2017 IEEE International Conference on Ubiquitous Computing and Communications (ISPA/IUCC)*. IEEE, 2017, pp. 331–3317.

[3] B. Li, C. Deng, J. Yang, B. Yuan, D. Lilja, and D. Du, "Haml-ssd: A hardware accelerated hotness-aware machine learning based ssd management," in *International Conference On Computer Aided Design*. IEEE/ACM, 2019.

[4] J. Park, Y. Jung, J. Won, M. Kang, S. Lee, and J. Kim, "Ransomblocker: a low-overhead ransomware-proof ssd," in *Proceedings of the 56th Annual Design Automation Conference 2019*. ACM, 2019, p. 34.

[5] B. Li and D. H. Du, "Tasecure: Temperature-aware secure deletion scheme for solid state drives," in *Proceedings of the 2019 on Great Lakes Symposium on VLSI*. ACM, 2019, pp. 275–278.

[6] P. Gutmann, "Secure deletion of data from magnetic and solid-state memory," in *Proceedings of the Sixth USENIX Security Symposium, San Jose, CA*, vol. 14, 1996, pp. 77–89.

[7] W. K. Weng and H. H. Wu, "Secure erase system for a solid state non-volatile memory device," Mar. 29 2012, uS Patent App. 12/891,631.

[8] H. A. Khouzani, C. Liu, and C. Yang, "Architecting data placement in ssds for efficient secure deletion implementation," in *Proceedings of the International Conference on Computer-Aided Design*. ACM, 2018, p. 42.

[9] M. Wang, J. Xiong, R. Ma, Q. Li, and B. Jin, "A novel data secure deletion scheme for mobile devices," in *2018 27th International Conference on Computer Communication and Networks (ICCCN)*. IEEE, 2018, pp. 1–8.

[10] J. Reardon, S. Capkun, and D. Basin, "Data node encrypted file system: Efficient secure deletion for flash memory," in *Proceedings of the 21st USENIX conference on Security symposium*. USENIX Association, 2012, pp. 17–17.

[11] W.-C. Wang, C.-C. Ho, Y.-H. Chang, T.-W. Kuo, and P.-H. Lin, "Scrubbing-aware secure deletion for 3-d nand flash," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 37, no. 11, pp. 2790–2801, 2018.

[12] D. Narayanan, A. Donnelly, and A. Rowstron, "Write off-loading: Practical power management for enterprise storage," *ACM Transactions on Storage (TOS)*, vol. 4, no. 3, p. 10, 2008.

[13] Y. Hu, H. Jiang, D. Feng, L. Tian, H. Luo, and C. Ren, "Exploring and exploiting the multilevel parallelism inside ssds for improved performance and endurance," *IEEE Transactions on Computers*, vol. 62, no. 6, pp. 1141–1155, 2013.