Secure File Deletion for Solid State Drives

Bhupendra Singh, Ravi Saharan, Gaurav Somani, Gaurav Gupta

Chapter 18

SECURE FILE DELETION FOR SOLID STATE DRIVES

Bhupendra Singh, Ravi Saharan, Gaurav Somani and Gaurav Gupta

Abstract Solid state drives are becoming more popular for data storage because of their reliability, performance and low power consumption. As the amount of sensitive data stored on these drives increases, the need to securely erase the sensitive information becomes more important. However, this is problematic because the tools and techniques used on traditional hard drives do not always work on solid state drives as a result of differences in the internal architectures. Single file sanitization is highly unreliable and difficult to accomplish on a solid state drive due to wear-leveling and related features. This chapter presents a reliable method for individual file sanitization on solid state drives. The method, called FTLSec, integrates a page-based encryption system in the generic flash translation layer. The efficacy of FTLSec is measured using a FlashSim solid state drive simulator. The results are compared with the well-known FAST flash translation layer scheme and an idealized page-mapped flash translation layer.

Keywords: Solid state drives, flash translation layer, secure file deletion

1. Introduction

Large amounts of sensitive data are stored in digital storage media. The disposal of this sensitive data becomes important when it is no longer needed or when the individuals or organizations desire to replace their outdated computing equipment. Secure data deletion is also a critical component of the disposal policy of government and commercial enterprises.

Secure deletion is the process of deleting data so that it becomes non-recoverable [14]. Secure deletion is also referred to as purging, sanitization and erasing. A number of tools, techniques and standards are available for secure deletion from traditional storage devices such as hard disk drives. However, since solid state drives (SSDs) are relatively new, fewer standards and techniques exist for this type of media.

Solid state drives have some advantages over mechanical spinning hard drives, but the flash memory they use has some inherent limitations such as finite erasure cycles [3] and erase-before-write [14]. Solid state drives store data in NAND flash memory, which is compartmented into blocks and pages. Each page contains multiple blocks. Pages are the units of read and write operations while erasures are performed on blocks. A page write must be preceded by an erasure. Because of the differences in atomicity, an erasure is a more costly operation than a read or write. Erasure also significantly degrades the overall write performance of flash memory – each block can be erased only a finite number of times (typically 10,000 to 100,000 times [3]) before it becomes unreliable.

To reduce the number of erasures, modern flash solid state drives include a flash translation layer (FTL). A flash translation layer uses a table that maps a logical address to a physical address to replace the erase-before-write by an out-of-place update method. When writing a new page, the flash translation layer selects an already free or erased page, writes to it, invalidates the previous version of the page and updates the mapping table. To implement this method, the flash translation layer employs a garbage collector (GC) [5, 7] to reclaim the invalid pages of a block by copying its valid pages (if any) to a new erased block and then erasing the entire original block. To lengthen the lifespan of a solid state drive, a flash translation layer implements a wear-leveling algorithm to distribute writes evenly across all the blocks in the flash solid state drive. Wear-leveling prevents data from being written continually to the same locations.

Solid state drives and hard disk drives have different internal architectures and storage mechanisms. As a result, the techniques applied to a hard disk drive for entire drive sanitization or single file sanitization may not work on a solid state drive. Secure deletion of a single file on a solid state drive is a relatively challenging task due to garbage collection and wear-leveling. The data that is left behind on invalid pages after an out-of-place update also complicates the secure deletion task.

To enable the secure deletion of a single file on a solid state drive, certain modifications to the generic flash translation layer are needed. This chapter describes a modification of the demand-based flash translation layer (DFTL) [5], which includes a page-based encryption system (PES). The modified flash translation layer technique is called FTLSec (flash translation layer with secure erase), which provides a means to securely delete a single file from a solid state drive. Experiments are conducted to demonstrate the utility of FTLSec and its superior performance compared with other flash translation layer schemes.

2. Background

Presenting an array of logical block addresses (LBAs) makes sense for a hard disk drive because its sectors can be overwritten. However, the approach is not suited to flash memory. For this reason, an additional component, namely the flash translation layer, is required to hide the inner characteristics of NAND flash memory and expose only an array of logical block addresses to the operating system. The flash translation layer resides in the solid state drive controller. The main function of the flash translation layer is address translation, in which the logical addresses that are seen by the operating system are converted to the physical addresses of NAND flash memory. The mapping table and the other data structures used by the flash translation layer are stored in a small, fast, on-flash static RAM (SRAM) based cache. The limited size of SRAM prevents the flash translation layer from achieving high performance. Furthermore, more data in a solid state drive, means that there is more data in the SRAM, which is undesirable because the price per byte of SRAM is higher than that of a solid state drive.

In addition to address translation, the flash translation layer implements garbage collection and wear-leveling. Garbage collection runs in the background to create free pages from invalid pages. Because a solid state drive has a limited number of erase cycles, wear-leveling is employed to increase the lifespan of the drive by attempting to distribute data so that every block has the same level of wear.

Flash translation layer schemes are classified into three types based on their mapping granularity: (i) page-level mapping; (ii) block-level mapping; and (iii) hybrid mapping. The three schemes are ordered roughly by their level of complexity. Each flash translation layer scheme offers trade-offs in terms of performance, memory overhead and life expectancy.

The page-level mapping scheme is a naive approach in which every logical page is mapped to a corresponding physical page. Therefore, if n logical pages are mapped to physical pages, the size of the mapping table is n. This scheme offers considerable flexibility, but it needs a large mapping table and, consequently, large SRAM, which can significantly increase manufacturing costs. For example, a 1 GB flash memory chip with a page size of 2 KB requires 2 MB of SRAM to store the mapping table [5].

The block-level mapping scheme divides logical addresses into logical blocks and offsets, and only the logical block addresses are mapped to the physical block addresses (PBAs). Block-level mapping can handle larger SRAMs than are feasible with page-level mapping. The logical block offset in a logical block is the same as the physical block offset in a physical block. This mapping requires 64 (number of pages/block) times less SRAM memory than the page-level mapping scheme.

A hybrid mapping scheme [10, 13] is a combination of the page-level and block-level mapping schemes. All hybrid flash translation layer schemes share a fundamental idea: log-block mapping, which uses an approach similar to log-structured filesystems. In this scheme, the physical blocks are divided into two groups: (i) data blocks; and (ii) log blocks. Data blocks constitute the major portion of a flash solid state drive and are mapped by a block-mapping scheme; a small fixed number of blocks are tagged as log blocks and are maintained with page granularity. When an update request to a page in the data block arrives, the hybrid log-block scheme writes new data to the log block allocated from the pool and invalidates the previous version of the data that was stored in the data block. Note that, for each write request, the logical address of the page is also stored in the out-of-band (OOB) area of each page.

3. Related Work

This section discusses existing work related to secure deletion on NAND flash drives. A user interacts with the physical medium using an interface. The interface offers functions that transform the user's data objects to a form suitable for storage on the physical medium. This transformation can also include operations such as encryption, error-correction and redundancy. Several layers and interfaces exist between user applications and the physical medium that stores data. Figure 1 shows the layers and interfaces through which data on a physical medium is accessed.

3.1 Layers and Interfaces

The lowest layer is the physical medium, also called the storage-management layer. NAND memory drives are accessed via a controller, specifically, an embedded processor that executes firmware code [14].

NAND flash memory is accessed via a flash translation layer controller that maps logical addresses from the operating system to the physical block addresses on the flash memory chip. Unlike a hard disk drive controller, the flash translation layer controller does not support in-place

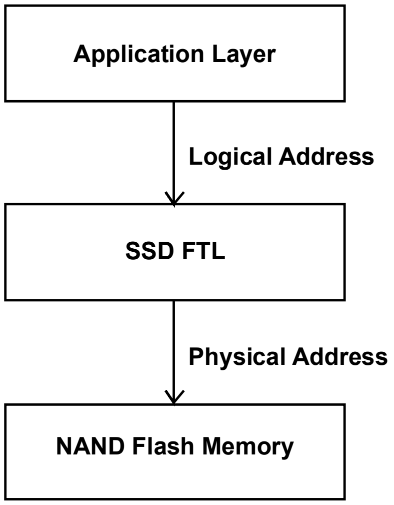


Figure 1. Interfaces involved in NAND flash memory data storage.

overwrites and remaps to new locations, leaving the stale data behind; this complicates secure deletion.

3.2 Choice of Layer for Secure Deletion

Secure deletion can be applied at any layer. Secure deletion at the lowest layer ensures that the data is securely deleted and is not recoverable; in contrast, secure deletion applied at the user layer allows the easy identification of data blocks. Secure deletion at the physical layer is preferable because the user does not know which data blocks to delete. Alternatively, a user layer approach allows for the easy identification of the data blocks to be deleted, but it has no information about the locations of the data blocks. A filesystem layer approach for secure deletion lies in between a physical layer approach and a user layer approach.

3.3 Controller Level Approaches

The flash translation layer manages the mapping between logical block addresses visible via standardized interfaces such as ATA and SCSI and physical pages of flash memory. These interfaces provide built-in erase commands that instruct on-board firmware to run a sanitization protocol on a flash drive. The commands are more reliable, but they are not supported by all flash drives. Moreover, the built-in commands are appropriate for entire drive sanitization, not for single file secure deletion.

The ATA interface specifications define the ERASE UNIT and ERASE UNIT ENH commands that securely erase all user-addressable areas [12]. The working draft ATA/ATAPI Command Set-3 (ACS-3) [18] incorporates a BLOCK ERASE EXT command that only succeeds if the sanitization feature set is supported by the device.

Wei et al. [19] have found that not all solid state drives support ATA commands. Some drives support ATA commands, but do not implement them correctly, leaving data intact on the drives. Other drives implement the ATA commands correctly. Thus, the support and implementations of ATA commands depend on the vendor. Swanson et al. [16] have proposed a reliable entire drive sanitization method that encrypts the data to be stored on a flash drive using a key, which is subsequently destroyed. In this case, every block is erased and written with a defined pattern, and erased again. Finally, the flash device is reinitialized by submitting a new (different) key to the flash controller.

3.4 Filesystem Level Approaches

Application level approaches are limited because they do not directly access the physical medium. Controller level approaches suffer from the inability to distinguish deleted data from valid data, because they cannot access the physical data locations and metadata. Filesystem level approaches lie in between these two approaches. The filesystem is generally unaware of the physical medium and accesses it via the device driver interface.

Data Compaction. Compaction balances the asymmetry between the write and erase granularities. It compacts a block (erase unit) containing the deleted data, copies the valid collocated data elsewhere and executes the erase operation. Copying data and then erasing an entire block is a costly operation that also adds wear to the NAND flash solid state drive. However, no immediate secure deletion approach based on erasure is available that can do better than one erase unit erasure per deletion. Immediate secure deletion requires a minimum of one erasure. Any improvement that further reduces the number of erase units erased per deletion must batch the deletions and perform intermittent secure deletion.

Single File Secure Deletion. Log-structured filesystems are commonly used as flash filesystems. Lee et al. [8] have proposed an efficient single file secure deletion approach for log-structured filesystems. In this approach, file data is encrypted and the keys are stored in the same block. To securely delete file data and metadata, the keys corresponding to the file on the block are simply erased. This approach was adapted to the YAFFS implementation [11].

Reardon et al. [14] have implemented a secure deletion approach for the UBIFS flash filesystem [6]. Each filesystem data node is encrypted when data on the data node is written and is decrypted when it is read from a flash memory device. Secure deletion is achieved by purging the keys from the logical key storage area, which houses all the keys.

Wei et al. [19] have proposed an immediate secure deletion approach for flash memory called scrubbing. Scrubbing re-programs individual pages to turn all the remaining ones into zeroes. The approach can remove data remnants by scrubbing pages that contain invalid data in a flash array, or it can prevent their creation by scrubbing the page containing the previous version when writing a new version. Erasing a flash memory erase unit is the only way to restore the charge to a cell, but cells can be drained when the write operation is used. These cells cannot be used to store new data, but their sensitive data is voided immediately. Fortunately, collocated data on the erase unit remains available. A concern with scrubbing is that it exhibits undefined behavior. Wei et al. have investigated the error rates for different types of memory and show that the rates vary widely. Scrubbing causes frequent errors in some devices and no errors in other devices.

3.5 Application Level Approaches

At the application layer, secure deletion approaches are carried out by user applications that interact with POSIX-compliant filesystems. The main approaches are: (i) overwriting the contents of a file before normal deletion (unlinking); (ii) unlinking a file and filling the free space on the drive; and (iii) overwriting the entire partition or drive by calling the secure deletion routine in the controller of the drive.

Overwriting makes multiple write passes with different bit patterns to sanitize a file [2]. Since a NAND flash drive does not allow in-place overwrites, the overwriting utilities do not necessarily work for securely deleting a single file.

Free space filling tools overwrite the free space on a drive and ensure that the sensitive data is securely deleted from the drive. Obviously, the cost of filling free space is proportional to the available free space. If the available free space is large, it will take a longer time to fill; this can result in high wear in flash memory devices.

Application level filling of free space is the only user level means of securely deleting data in YAFFS [14]. Reardon et al. [14] has introduced two methods that work on the YAFFS filesystem: (i) purging; and (ii) ballooning. Purging erases free space immediately while ballooning performs frequent secure deletions by keeping the filesystem full of junk data, causing more frequent YAFFS garbage collection.

Erasing the entire drive securely deletes data stored in NAND flash memory, including metadata and directory structures. The time required for this process depends on the size of the drive; erasure can take a long time for a large-capacity drive.

3.6 Cross Layer Approaches

As discussed above, secure deletion at the lowest layer has no information about the locations of deleted data objects. Thus, the higher layers can be used to pass information about the deleted data to the lowest layer, permitting the secure deletion of deleted data objects. TRIM [1] and TrueErase [4] use this type of approach.

A TRIM command [1] enables the operating system to wipe the pages that contain invalid data due to deletions by the user or the operating system. Trimming allows the solid state drive to handle garbage collection overhead that would otherwise significantly slow down future write operations on the involved blocks. TRIM wipes out the invalid pages on demand, so it ensures the secure deletion of data. TRIM commands were not designed for secure erasure, but instead to accelerate write operations on flash media. However, it is not possible to restrict TRIM commands only to sensitive blocks, which means that the underlying mechanism that securely deletes data must be efficient. TRIM commands are only effective when they are supported by the operating system and by the solid state drive.

TrueErase [4] irrevocably erases data and metadata on hard disk drives and solid state drives. TrueErase is a per-file, encryption-free secure deletion approach that keeps track of the sensitive data throughout the storage path. Information about the deleted blocks is forwarded to the lowest layer of the new communications channel added between the device driver and filesystem. To ensure secure deletion, the device driver is modified to implement immediate secure deletion using its lower layer interface. TrueErase is more effective than TRIM commands because it deletes data at a smaller granularity without any delayed deletions.

4. Proposed Secure Deletion Approach

The proposed secure deletion approach uses a page-based encryption system (PES) to securely erase files. The approach also incorporates some modifications to the generic flash translation layer. The approach involves three steps: (i) encrypt each page before writing it to the NAND

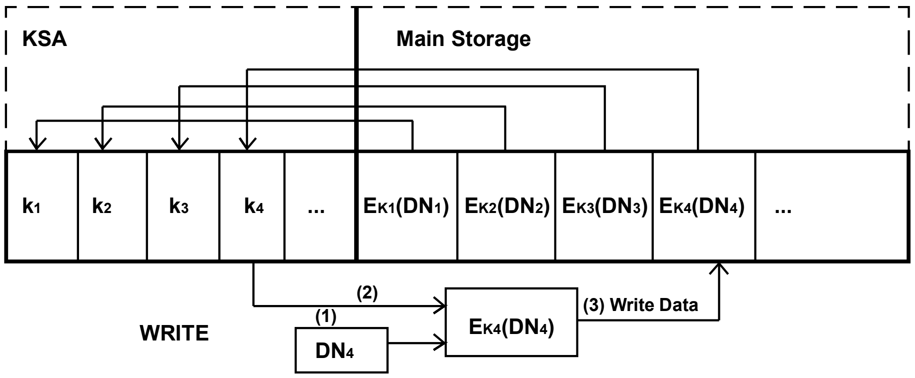


Figure 2. Writing to a new page.

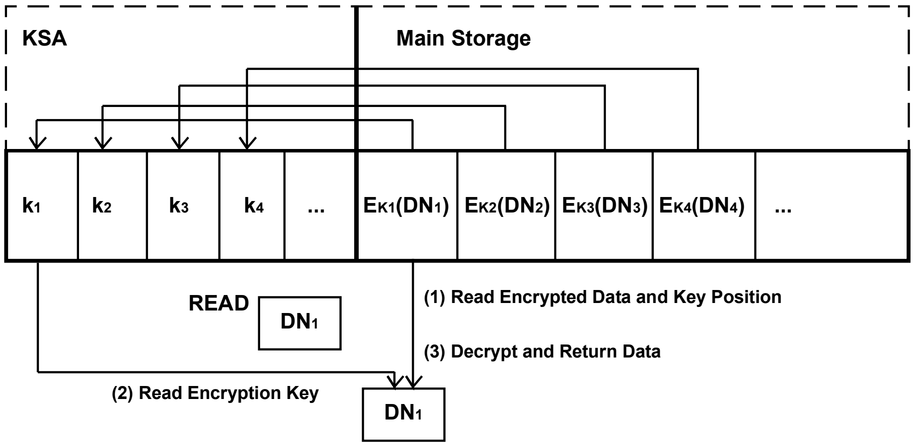


Figure 3. Reading from Page DN_1.

flash and decrypt each page before reading the data; (ii) store the keys in a fixed area called the key storage area (KSA); and (iii) when a specific file is to be deleted, erase the block where the file encryption keys are stored. Figure 2 shows data being written to a new page while Figure 3 shows data being read from a page.

Key Storage Area. The key storage area is reserved. Migrating blocks of the NAND flash solid state drive are used to store the encryption/decryption keys of all the pages. The locations of the encryption/decryption keys of a page are stored in the page header. The locations correspond to the logical key storage area number and offset. In order to delete a key (which decrypts deleted data), the blocks in the key storage area must be erased regularly. When a page becomes invalid upon removal or updating, the encryption/decryption keys of the page that are maintained in the key storage area are marked as deleted. This approach is different from file deletion – whenever a page is discarded, its corresponding encryption/decryption keys are marked as deleted. A deleted key remains tagged as deleted until it is purged and a fresh random key takes its place; this fresh key is tagged as unused.

Purging. Purging is a regular process that erases keys from the key storage area. Purging is done on each block of the key storage area. In this process, a new block of key storage area is selected to copy the used keys that reside in the same locations, and the deleted or unused keys are replaced with new unused random data. This random data, which is inexpensive and easy to generate, is assigned to the new keys when needed.

Keys that are tagged as unused are logically-fixed in the key storage area because their corresponding pages are already stored on the NAND flash solid state drive until erase operations are performed at the key locations. All the blocks containing invalid data are erased, thus purging the unused and deleted keys along with the pages they encrypt.

Key State Map. The key state map maps key positions to key states. Keys can be in one of three states – unused, used or deleted. Unused keys are keys that can be assigned and then tagged as used. Used keys are used to encrypt/decrypt valid pages; these keys ensure the availability of user data. Deleted keys are keys that are used to encrypt/decrypt deleted data, specifically, pages that are no longer needed by the filesystem and should be securely erased by the filesystem to achieve the goal of secure deletion. A purging operation replaces unused and deleted keys with new values; the used keys remain on the storage medium.

Figures 4 and 5 show representations of key state maps before aand after keys are purged.

Flash Translation Layer Modifications. In order to enable the secure deletion of a single file, the generic flash translation layer is modified and a page-based encryption system is integrated in it. All the incoming data is encrypted before writing to NAND flash memory and decrypted for each read operation. The keys are stored in the reserved key storage area and key locations are assigned to the data of each page. The page-level mapping table of the modified flash translation layer also stores a reference to the encryption/decryption key positions so that the

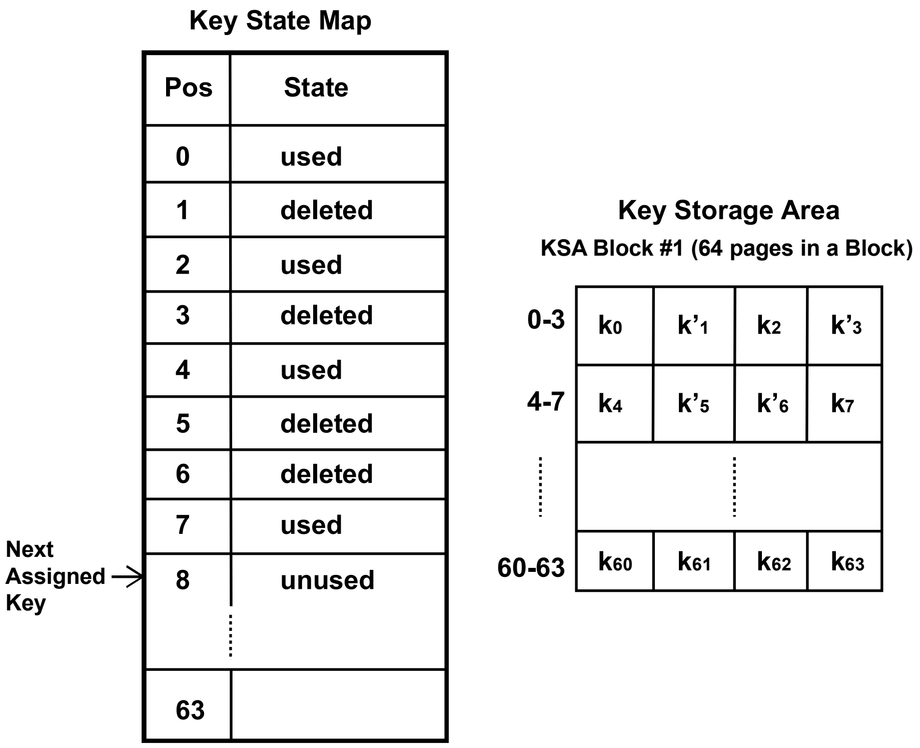


Figure 4. Key state map before purging keys.

algorithm that builds its logical pages to physical pages, also builds the corresponding encryption/decryption key positions.

A key position has two parts: (i) logical key storage area block number; and (ii) offset within the key storage area block. The metadata relating to each key storage area block, comprising the logical key storage area block number and epoch number, are stored in the last page of each logical key storage area block.

5. Experimental Results

No publicly-available flash solid state drive prototype exists for testing flash translation layer schemes. Therefore, experiments were performed using an open-source, validated flash solid state drive simulator named FlashSim [7]. The simulator was used to evaluate various flash translation layer schemes proposed in the literature. FlashSim uses a modular architecture and facilitates the integration of new flash translation layer schemes. In the experiments, FlashSim was used to evaluate the performance of the proposed flash translation layer (FTLSec) along with two

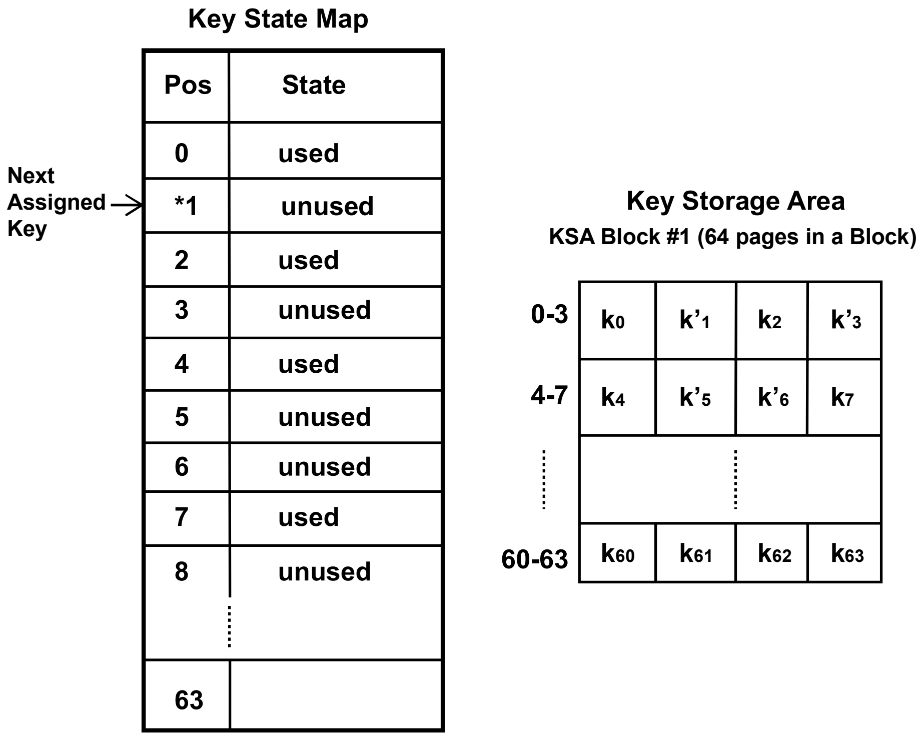


Figure 5. Key state map after purging keys.

traditional flash translation layer schemes: (i) a hybrid flash translation layer scheme (FAST) [9]; and (ii) an ideal page-mapped flash translation layer scheme [5].

5.1 Experimental Setup

The experiments simulated 16 GB of NAND flash memory. Table 1 presents the simulation parameters. The number of extra blocks was varied from 5% to 9% of the total blocks. The extra blocks were used as log blocks by the FAST hybrid flash translation layer scheme.

Experimental Workloads. Real-world traces were used as workloads to measure the performance of FTLSec against the hybrid flash translation layer scheme FAST and the idealized page-mapped flash translation layer scheme. Different types of traces that have been used to evaluate the performance of storage systems were selected.

Table 2 lists the characteristics of each type of trace. Financial1 was taken from OLTP applications running at two large financial in-

Table 1. Simulation parameters.

Parameter	Value
SSD Capacity	16 GB
Page Size	2 KB
Page OOB	64 B
Pages per Block	64 pages
Percentage of Extra Blocks	5% to 9%
Page Read Delay	0.1309 ms
Page Write Delay	0.4059 ms
Block Erase Delay	2 ms

Table 2. Experimental workloads.

Workload	Average Request Size (KB)	Read (%)	Write (%)
Financial1	3.00	9.0	76.8
WebSearch	14.86	97.3	2.70
Exchange	12.00	46.4	53.6

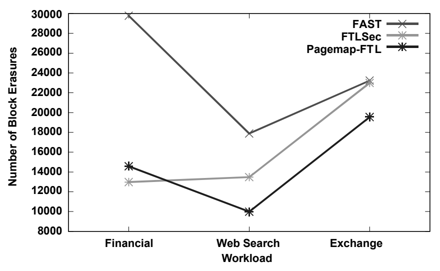
stitutions (OLTP Application I/O) [17]. The Financial1 trace is write-dominant. The WebSearch trace is a read-dominant I/O trace (Search Engine I/O) [17]. The Exchange trace was collected from a Microsoft Exchange mail server [15].

Performance Metrics. Two metrics were used in the simulations: (i) number of erased blocks (indicator of garbage collection efficacy); and (ii) average response time (device service time plus the time spent in the driver queue).

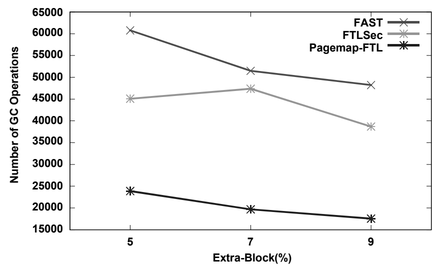
5.2 Garbage Collection Overhead

Garbage collection may involve various types of merge operations (e.g., switch, partial and full) at the time of servicing update requests. These merge operations create overhead in the form of block erases and when copying valid pages in the blocks (victim blocks) to other (free) blocks.

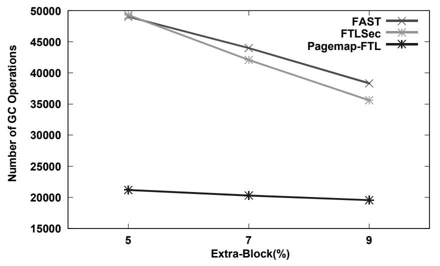
Figure 6 shows the impact of the garbage collection overhead. The results demonstrate that FTLSec has fewer garbage collection operations than FAST for all the workloads. Also, when the number of extra blocks used for log blocks increases, the number of garbage collection operations decreases. Figure 6 also shows that the number of block erasures is the



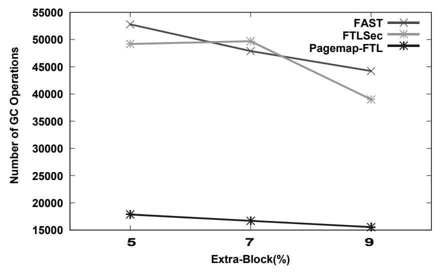
(a) Block erasures.



(b) Financial1 traces.



(c) WebSearch traces.



(d) Exchange traces.

Figure 6. Garbage collection overhead.

lowest for the WebSearch workload. This is because WebSearch is highly read-dominant and issues a small number of write requests. As a result, WebSearch has a lower garbage collection overhead than the Financial1 and Exchange traces.

5.3 Impact of Extra Blocks

As discussed above, extra blocks are used to support merge operations during garbage collection. The number of extra blocks in the experiments was varied from 5% to 9% of the total blocks while the solid state drive capacity was kept constant throughout the experiments.

6. Countering Secure Deletion

To counter anti-forensic approaches such as secure deletion on solid state drives, a digital forensic investigator must have a good understanding of the tools and techniques used to securely delete drive content. This section highlights some anti-anti-forensic techniques to counter secure deletion when one or more files are securely deleted and when an entire drive is sanitized.

6.1 Countering Single File Sanitization

The targets of the most privacy protection tools are messenger chats, pictures, documents and video files. Secure removal of these artifacts with overwriting techniques does not guarantee their complete removal and leaves some traces. The traces include:

- **Recent Events:** Recent events such as browser histories, Skype and Hangout chats, documents and downloads can still be found in the *Pagefile.sys* and *Hiberfil.sys* files.

- **Multiple Copies:** Windows often creates multiple copies of some types of files. These copies are generated when copying or moving files, temporarily saving copies of working documents, and compressing and decompressing files. Performing a secure deletion only on the target file leaves the other copies intact, which means that they could be located and extracted by digital forensic investigators.

- **File Fragments:** If a file on a solid state drive is erased by overwriting, fragments of the file may still reside on the drive because most operating systems have filesystem fragmentation. It is possible to partially carve a file if the fragments left before defragmentation have not been overwritten.

- **Volatile Evidence:** A computer or laptop that was running when it was seized may still have artifacts in RAM.

- **Volume Shadow Copy:** Older versions of a file may exist in the volume shadow copy.

6.2 Countering Entire Drive Sanitization

The sanitization of an entire solid state drive is performed with vendor-defined commands that, in turn, invoke TRIM. Modern solid state drives are compliant with DRAT (definite read after trim) or DZAT (definite zero after trim) [1]. This makes the solid state drive controller return all zeroes or garbage data, even if the drive contains actual data. In this case, the recovery of solid state drive data is a difficult and time-consuming task. However, if the solid state drive firmware is not updated or is buggy, the data may still be found on the disk.

7. Conclusions

Solid state drives with NAND flash memory are becoming increasingly common for storing sensitive data. This makes the sanitization of solid state drives a critical component of data management. The FTLSec (file translation layer with secure erase) method presented in this chapter reliably sanitizes individual files on a solid state drive. FTLSec achieves its functionality by integrating a page-based encryption system in the generic flash translation layer. Experiments demonstrate the FTLSec has fewer block erasures and garbage collection operations compared with the well-known FAST flash translation layer scheme and an idealized page-mapped flash translation layer scheme.

References

[1] O. Afonin, D. Nikolaev and Y. Gubanov, Countering anti-forensic efforts – Part 2, *Forensic Magazine*, September 16, 2015.

[2] Australian Signals Directorate, Information Security Manual, Kingston, Australia (`www.asd.gov.au/infosec/ism/index.htm`), 2015.

[3] T. Chung, D. Park, S. Park, D. Lee, S. Lee and H. Song, A survey of the flash translation layer, *Journal of Systems Architecture*, vol. 55(5-6), pp. 332–343, 2009.

[4] S. Diesburg, C. Meyers, M. Stanovich, M. Mitchell, J. Marshall, J. Gould, A. Wang and G. Kuenning, TrueErase: Per-file secure deletion for the storage data path, *Proceedings of the Twenty-Eighth Annual Computer Security Applications Conference*, pp. 439–448, 2012.

[5] A. Gupta, Y. Kim and B. Urgaonkar, DFTL: A flash translation layer employing demand-based selective caching of page-level address mappings, *ACM SIGPLAN Notices*, vol. 44(3), pp. 229–240, 2009.

[6] A. Hunter, A Brief Introduction to the Design of UBIFS, Version 0.1 (`www.linux-mtd.infradead.org/doc/ubifs_whitepaper.pdf`), 2008.

[7] Y. Kim, B. Tauras, A. Gupta and B. Urgaonkar, FlashSim: A simulator for NAND-flash-based solid-state drives, *Proceedings of the First Conference on Advances in System Simulation*, pp. 125–131, 2009.

[8] J. Lee, S. Yi, J. Heo, H. Park, S. Shin and Y. Cho, An efficient secure deletion scheme for flash filesystems, *Journal of Information Science and Engineering*, vol. 26(1), pp. 27–38, 2010.

[9] S. Lee, D. Park, T. Chung, D. Lee, S. Park and H. Song, A log-buffer-based flash translation layer using fully-associative sector translation, *ACM Transactions on Embedded Computing Systems*, vol. 6(3), article no. 18, 2007.

[10] S. Lee, D. Shin, Y. Kim and J. Kim, Last: Locality-aware sector translation for NAND flash memory based storage systems, *ACM SIGOPS Operating Systems Review*, vol. 42(6), pp. 36–42, 2008.

[11] C. Manning, How YAFFS Works (`www.yaffs.net/sites/yaffs.net/files/HowYaffsWorks.pdf`), 2012.

[12] P. McLean (Ed.), Information Technology – AT Attachment-3 Interface (ATA-3), X3T13 2008D Revision 7b, X3T13 Technical Committee, American National Standard of Accredited Standards Committee X3, Washington, DC (`www.scs.stanford.edu/11wi-cs140/pintos/specs/ata-3-std.pdf`), 1997.

[13] C. Park, W. Cheon, J. Kang, K. Roh, W. Cho and J. Kim, A reconfigurable FTL (flash translation layer) architecture for NAND flash based applications, *ACM Transactions on Embedded Computing Systems*, vol. 7(4), article no. 38, 2008.

[14] J. Reardon, D. Basin and S. Capkun, SoK: Secure data deletion, *Proceedings of the IEEE Symposium on Security and Privacy*, pp. 301–315, 2013.

[15] Storage Networking Industry Association, SNIA IOTTA Repository, Microsoft Enterprise Traces, Colorado Springs, Colorado (`iotta.snia.org/traces/130`), 2011.

[16] S. Swanson and M. Wei, SAFE: Fast, Verifiable Sanitization for SSDs, Non-Volatile Systems Laboratory, Department of Computer Science and Engineering, University of California – San Diego, San Diego, California (`cseweb.ucsd.edu/~swanson/papers/TR-cs2011-0963-Safe.pdf`), 2010.

[17] UMass Trace Repository, Storage, Laboratory for Advanced Software Systems, University of Massachusetts, Amherst, Massachusetts (`traces.cs.umass.edu/index.php/Storage/Storage`), 2007.

[18] R. Weber (Ed.), Information Technology – ATA/ATAPI Command Set-3 (ACS-3), T13/2161-D Revision 5, T13 Technical Committee, American National Standard of Accredited Standards Committee INCITS, Washington, DC (`www.t13.org/Documents/UploadedDocuments/docs2013/d2161r5-ATAATAPI_Command_Set_-_3.pdf`), 2013.

[19] M. Wei, L. Grupp, F. Spada and S. Swanson, Reliably erasing data from flash-based solid state drives, *Proceedings of the Ninth USENIX Conference on File and Storage Technologies*, 2011.