# Secure File Delete in NAND-based Storage

Ilhoon Shin

*Electronic and IT Media Engineering Department,*
*Seoul National University of Science and Technology*
*ilhoon.shin@snut.ac.kr*

***Abstract***

*The existing secure file delete tool distorts secure contents by over-writing the randomly generated data many times, which decreases the recoverability of the secure contents. However, in NAND-based storages, the original secure contents are preserved even after performing the repetitive over-write requests, which is because of the out-of-place update of flash translation layer. In order to address the problem, this paper studies the method to implement the secure file delete as to the representative flash translation layer schemes. The result shows that the secure file delete can be implemented in the block mapping scheme or in the BAST scheme.*

***Keywords:*** *secure file delete, flash translation layer, NAND flash memory*

## 1. Introduction

NAND flash memory has been dominating the mobile storage market with the strengths of light-weight, low power consumption, silence, shock-resistance, and small form factor. Nowadays, NAND flash memory is encroaching on the share of hard disk drives even in laptop, PC, and server markets, with the better performance and the competitive price and capacity. Lots of previous researches to improve the performance of NAND-based storage devices have contributed the success of NAND flash memory. However, security issues when using NAND-based devices have not been sufficiently investigated. This paper focuses on implementing secure file delete in the NAND-based storage devices.

When deleting a file, the general file systems such as EXT3, FAT, or NTFS modifies the contents of the parent directory and the metadata such as block allocation information. For a better performance, the actual contents of the deleted files are preserved without modification. Therefore, the deleted contents can be recovered easily by reading the disk sectors directly.

In order to prevent the deleted secure contents from being recovered by illegal hackers, the secure file delete tools distort the original information by overwriting the existing file or disk area with the randomly generated number [4]. The overwritten area is hard to be recovered because the original contents were distorted. However, in NAND-based storage devices, the original contents are preserved even after the secure file delete tools overwrite the target file many times. This is because of the characteristics of flash translation layer (FTL) which is deployed in the NAND-based storage devices.

NAND flash memory is a kind of EEPROM (Electrically Erasable Programmable Read Only Memory) that does not support the overwrite operation. Once a cell is programmed (written), it cannot be re-written. In order to write new data, the cell should be erased first. Because erasing a cell is order of magnitude slower than the read/write operations and erasing unit is larger than the unit of read/write operations, it is almost impossible to implement an in-place update which writes the new data to the original physical position. Instead, NAND-based storage devices implement an out-of-place update which writes the new data to another clean cell. The original cell is marked as invalid. In the out-of-place update, the original content is preserved even when the file system sends the over-write requests, and thus the secure contents can be recovered by reading the cells directly.

In order to prevent secure contents from being recovered, the original cell should be erased by the overwrite requests of the file systems. This paper investigates the method to implement the secure file delete as to the representative FTL schemes.

The rest of the paper is organized as follows. Section 2 explains the representative FTL schemes. Section 3 investigates the possible way to implement the secure file delete. Finally, section 4 draws a conclusion and discusses future work.

## 2. Flash Translation Layer

NAND flash memory consists of blocks and pages. A block is a unit of erase operation and consists of multiple pages. A page is a unit of read/write operations. As described in the section 1, NAND flash memory does not support the over-write operations, and thus NAND-based storages deploy FTL which emulates the block device interfaces by performing the out-of-place update. In the out-of-place update, the physical location of valid data becomes different on every write request, and therefore FTL maintains the mapping information between logical sector numbers and their current physical locations. According to the unit of the mapping information, FTL is classified to page mapping, block mapping, and hybrid mapping schemes.

In the page mapping scheme [5], data are written in a NAND page unit, and the mapping unit is a NAND page. On the write requests, FTL searches for clean pages for the new data and writes the data to the found clean pages. The obsolete pages are marked as invalid and the mapping table is updated. Note that the previous contents of the obsolete pages are not modified by the over-write requests. The page mapping scheme delivers a good performance. However, it consumes large memory to maintain the mapping table.

The block mapping FTL [6] writes the data in a NAND block unit and maintains the mapping table in a block unit. On the write requests, FTL searches for a clean block and writes the data to the found block together with the unmodified data of the old block. The old block is marked as invalid. Note that the previous contents are also unmodified by the over-write requests. The block mapping scheme reduces the memory consumption much. However, the performance is hurt by the excessive copy overhead of the unmodified data.

In order to address the drawbacks of the page mapping scheme and the block mapping scheme, the hybrid mapping schemes were presented. The BAST (Block Associative Sector Translation) scheme [1] is one of representative hybrid mapping schemes. The BAST scheme uses several NAND blocks as write buffer. The blocks that are used as write buffer are called log blocks and the other blocks are called data blocks. The BAST scheme operates the block mapping scheme for the data blocks and the page mapping scheme for the log blocks. On the write requests, FTL searches for the associated log block with the target data block, and the data are written to the found log block. The log blocks absorb small sized write requests and thus the BAST scheme delivers a better performance than the block mapping scheme. However, it is vulnerable to the random write pattern because a log block can be associated with only one data block and the random write pattern exhausts the log blocks frequently.

The FAST (Fully Associative Sector Translation) scheme [2] allows the log blocks shared by multiple data blocks to cope with the random write pattern. On the write requests, the data are written to the current working log block regardless of the logical sector numbers. If the current log block does not have clean pages, the next log block becomes the working log block. The FAST scheme utilizes the log block space fully and delivers a better performance than the BAST scheme in the random write pattern. However, the computation overhead of finding the location of valid sector is huge because the valid sectors can be distributed over all the log blocks. Shin presented to use the hashed page table to reduce the computation overhead of the FAST scheme [3].

## 3. Implementation of Secure File Delete in NAND-based Storages

In this chapter, we investigate the feasibility of implementing secure file delete as to the representative FTL sector mapping schemes described in the section 2 and describe its implementation if it is feasible.

In the page mapping scheme, the obsolete data invalidated by the overwrite requests remains until the garbage collection time. The secure contents are completely deleted only if the garbage collector selects the NAND block that contains the secure contents as victim. The problem is that the time that the garbage collector is initiated is non-deterministic. If there are lots of clean pages, the garbage collector is not initiated even when the file systems send the over-write requests many times. In addition, other block can be selected as victim by the garbage collector. Therefore, implementing the secure file delete under the page mapping scheme is almost impossible. In order to implement it, a special command for the secure delete should be added to the standard block interface such as SCSI.

Meanwhile, the block mapping scheme writes new data to a clean block together with the unmodified data on the over-write requests. At this time, the old block is invalidated. Thus, the secure file delete can be easily implemented by erasing the invalidated block immediately on the over-write requests. Different from the page mapping scheme, the time that the obsolete data are erased is deterministic, which is done during handling over-write requests. Therefore, the secure file delete tool can guarantee that the secure contents are completely removed with the over-write requests. The drawback of this implementation is a low performance, which stems from the limitation of the block mapping scheme. The overall performance is seriously damaged in the block mapping scheme because of the excessive copying overhead of the unmodified data.

In the BAST scheme, the obsolete data invalidated by the overwrite requests also remain until the garbage collection time. For example, let’s assume that the secure data were written in the first page of the associated log block and the rest pages of the log block are clean. If the secure file delete tool sends the over-write request to the secure data, the randomly generated data are written to the second page of the associated log block and the first page that contains the secure data is invalidated. The secure data are preserved with unmodified. In order to completely erase the secure data, the secure file delete tool should send the over-write requests at least the same times with the number of pages in the NAND block. The repetitive over-write requests will exhaust the clean pages in the associated log block and initiate the garbage collector. The garbage collector merges the data block and the associated log block, which erases all the invalidated data including the target secure data. Therefore, the secure file delete tool can guarantee that the secure contents are completely removed by sending the repetitive over-write requests. The limitation of this implementation is that the secure file delete tool should know the number of NAND pages in a NAND block in the underlying NAND-based storages. Another drawback of this implementation is that it delivers a low performance in the random write pattern, which stems from the limitation of the BAST scheme.

In the FAST scheme, the obsolete data invalidated by the overwrite requests remains in the log block until the garbage collection time. Similarly to the page mapping scheme, the secure data can be completely removed only if the garbage collector selects the log block that contains the secure contents as victim. Thus, the time that the secure data is completely deleted is non-deterministic, and the secure file delete tool cannot guarantee the complete removal of secure data even with the repetitive over-write requests. Therefore, implementing the secure file delete under the FAST scheme is also almost impossible. In order to implement it, a special command for the secure delete should be added to the standard block interface similarly to the page mapping scheme.

## 4. Conclusion

Conclusively, the secure file delete can be implemented under the block mapping scheme or under the BAST scheme. Especially, the secure file delete is easily implemented in the block mapping scheme by instantly erasing the obsolete block on the over-write request. When using the page mapping scheme or the FAST scheme, the complete removal of secure data is not guaranteed.

The limitation of the presented solution is the low performance, which stems from the characteristics of the block mapping scheme and the BAST scheme. We need to design a new sector mapping scheme which is able to achieve a good performance while at the same time guaranteeing the secure file delete. Another alternative is to add a special command to the standard block device interface for the secure file delete.

## Acknowledgements

This work was supported by Seoul National University of Science and Technology and by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education, Science and Technology (2010-0003938).

## References

[1] J. Kim, J. M. Kim, S. Noh, S. Min and Y. Cho, “A space-efficient flash translation layer for compact flash systems”, IEEE Transactions on Consumer Electronics. Vol. 48, pp. 366-375 **(2002)**.

[2] S. Lee, D. Park, T. Chung, W. Choi, 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, No. 3 **(2007)**.

[3] I. Shin, “Reducing computational overhead of flash translation layer with hashed page tables”, IEEE Transactions on Consumer Electronics, Vol. 56, pp. 2344-2349 **(2010)**.

[4] P. Gutmann, “Secure deletion of data from magnetic and solid-state memory”, Proceedings of USENIX Security Symposium, **(1996)**.

[5] A. Ban, “Flash file system”, U.S. Patent 5,404,485 **(1995)**.

[6] A. Ban, “Flash file system optimized for page-mode flash technologies”, U.S. Patent 5,937,425 **(1999)**.

## Authors

**Ilhoon Shin**

Ilhoon Shin received the B.S., the M.S., and the ph.D degrees in computer science and engineering from Seoul National University, Korea. He is currently an assistant professor of the department of electronics and information engineering at Seoul National University of Science & Technology. His research interests include storage systems, embedded systems, and operating systems.