A Write-Friendly NVM Scheme for Security Metadata with High Availability

Jianming Huang and Yu Hua
School of Computer, Huazhong University of Science and Technology
Email: {jmhuang, csyhua}@hust.edu.cn

I. Introduction

Non-Volatile Memory (NVM) is becoming the main devices of next-generation memory systems, due to high density, near-zero standby power, non-volatile and byte-addressable features. However, due to the non-volatile property, NVM has to handle severe security vulnerabilities. After physically stealing DIMM, an attacker can easily read the contents via another computer due to retaining data after power off in NVM. To protect the user data from attacks, an intuitive solution is to leverage data encryption and integrity verification schemes, which unfortunately introduce extra security metadata, respectively including counter blocks and integrity tree nodes. To support the execution of applications after crashes, it is important to efficiently recover these metadata to consistent states.

To ensure data security in NVM, existing security mechanisms, i.e., counter mode encryption (CME) [2] and integrity tree verification [5], are leveraged in the memory controller. CME uses a counter in counter blocks, a data line address and a global secret key to generate a one-time padding (OTP) via the AES algorithm. For memory writes, the data line needs to be encrypted by XORing the plaintext data and OTP. For memory reads, OTP is generated via the cached counter block in parallel with reading a memory line, and the plaintext data are obtained by XORing the memory line and OTP. Thus the decryption latency is hidden by the latency of reading data. SGX-style integrity tree (SIT) is widely used in the integrity verification scheme. An SIT node contains eight 56-bit counters and one 64-bit MAC space [5]. The MAC in each SIT node is generated by hashing the node address, the counters in this node and one corresponding counter in the parent node. SIT calculates the MACs in different nodes in parallel once these counters have increased. In the SIT-based systems, the counter blocks in CME are the leaves of SIT, and the counter blocks have the same structures as SIT nodes, i.e., eight counters and one MAC.

In DRAM, the data are lost upon system crashes. Unlike DRAM, NVM maintains data after system crashes and provides the ability to recover the system. However, the security metadata, i.e., SIT nodes and counter blocks, in the metadata cache in the memory controller are lost. To continue protecting the data in NVM, the security metadata also need to be recovered during system recovery. Reconstructing one SIT node needs its parent node as inputs. Unfortunately, the parent node in cache may lose upon system crashes. Therefore, the SIT cannot be reconstructed after system recovery. Moreover, for high-availability systems that need to meet the availability target of 99.999% (five nines rule), such as bank systems and online transactions, a short recovery time is necessary.

In SIT, when persisting data, the parent node of the persisted data is modified. To recover SIT after system crashes, Anubis [5] stores the modifications of security metadata in a shadow table (ST) block and persists the ST block. Therefore, after system crashes, the stale metadata in NVM are restored from the ST blocks. However, Anubis incurs 2x memory writes, i.e., for normal writes and ST block writes, which significantly reduces the system performance and NVM lifetime, and increases the energy overheads.

To achieve a short recovery time after system crashes and low write overhead on running time, we propose STAR to instantly persist modifications of metadata and fast recover the stale metadata on recovery. The insight behind STAR is that the modifications in metadata cache are caused by persisting one data. To avoid two writes respectively for the data and modification, we coalesce them into one write maintained in the data to be persisted.

II. STAR Design

A. Observation

SIT uses MACs stored in tree nodes to associate one tree node with its parent node. The user data also need MACs to associate the data with encryption counters by hashing the data contents, data address and corresponding counter in counter block to generate the MAC. To avoid the failure of integrity checking after recovery, MACs are written into NVM with the user data [4].

In general, the size of MAC space is 64 bits, and 56-bit MAC is stored in the MAC space [5]. In fact, 54-bit MAC is enough to guarantee security [3]. There are 10 unused bits in a 64-bit MAC space. We will reuse these unused bits in our design to instantly persist the modifications of security metadata.

B. Persisting the Modifications of Metadata

In SIT, when one data block is evicted from cache into NVM, its parent nodes are modified. Specifically, the corresponding counter in its parent node increases by one, and the MAC in the parent node is hence modified. Other counters

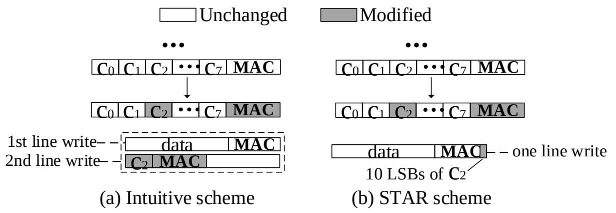

Fig. 1. Persisting the modifications of dirty nodes. (a) The intuitive scheme persists two lines with atomicity assurance; (b) STAR scheme persists one line via counter-MAC synergization.

in the parent node and other nodes are unchanged as shown in Fig. 1. The systems need to persist the modifications. On recovery, after obtaining the modified counters of the stale nodes, we restore the node by combining the modifications and the stale version in NVM. Finally, the MAC of the stale node is recomputed.

To persist modifications of the parent node, one intuitive scheme is to store the modified counter and MAC in one line, and persist the line with data block in an atomic operation, as shown in Fig. 1(a). However, the intuitive scheme incurs 2x memory writes.

Unlike the intuitive scheme, we store the modifications of metadata in their child nodes' MAC space, as shown in Fig. 1(b). STAR leverages the unused bits of MAC in the child node to store 10 LSBs of the corresponding counter in the parent node. When one data to be written arrives at the memory controller, only the corresponding counter in the parent node increases by one. STAR stores 10 LSBs of the increased counter in the unused space of the data's MAC, called *counter-MAC synergization*. In data, the MAC and contents are organized in one line [4]. Hence the modifications of the parent node are atomically persisted with the data without any atomicity assurance.

To protect the LSBs, MAC in a data block is computed by hashing the contents in the block, the address of the block, the corresponding counter in the parent node and the LSBs stored in the MAC space. When a counter in one metadata has been increased 2^{10} times, the metadata needs to be flushed into NVM to update the Most Significant Bits (MSBs), which are used on recovery to restore the stale metadata. This counter overflow is rare and introduces negligible overhead.

After system crashes, the counter blocks are restored from the corresponding user data which are the child nodes of counter blocks, and the SIT nodes are restored from their child nodes. For one stale metadata, STAR obtains the correct LSBs of counters from its child nodes' MAC space. Combining the MSBs stored in NVM with the LSBs obtained from the child node's MAC, counters in the stale metadata are restored. According to the corresponding counter in the parent node (if necessary, the parent node also needs to be restored), the MAC in this stale metadata is recomputed. By using the counters and MACs, the stale metadata are recovered.

III. Experimental Results

To evaluate the performance of STAR, we use Gem5 with NVMain to model the system and compare our STAR with a persistent system using a write-back metadata cache (WB), a strict persistence scheme (Strict Persistence) and Anubis for SGX-style Integrity Tree scheme (Anubis).

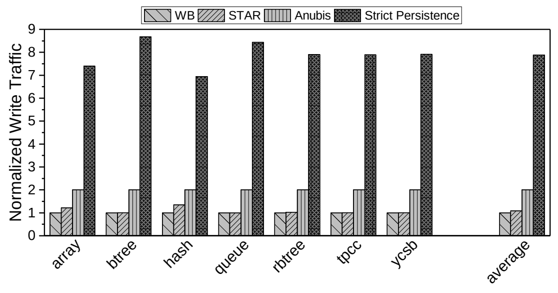

Fig. 2. The write traffic of different schemes (normalized to WB).

Fig. 2 shows the write traffic of different schemes. In addition to persisting common memory writes (i.e., the writes in WB), Anubis also persists the ST blocks, and the strict persistence persists all nodes in a branch of SIT when a user data is written. Since the height of SIT is 8 in our configuration (excepting the root), the write traffic (including persisting the user data) of a strict persistence scheme is about 8 times higher than that of WB. As shown in Fig. 2, compared with the baseline WB scheme, the write traffic of STAR is 1.08x, while Anubis has 2x write traffic than WB. STAR significantly reduces 92% extra memory traffic compared with Anubis.

In the full-length paper [1] we describe STAR in more detail: threat model; fast recovery; security analysis; the IPC performance and energy consumption of STAR.

IV. Conclusion

To ensure the data security in NVM, the security metadata need to be recovered, which requires elaborate design for memory writes during running time. In this paper, we propose STAR, which stores the modifications of security metadata in the MAC space of persisted data. Therefore, STAR atomically persists the modifications and data without extra memory writes. To efficiently recover the SIT and verify the recovery process, STAR leverages bitmap lines to maintain the locations of stale metadata. A cache-tree is constructed to ensure the correctness of the recovery process.

References

[1] J. Huang and Y. Hua, "A write-friendly and fast-recovery scheme for security metadata in non-volatile memories," in *The 27th IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, 2021. [Online]. Available: https://doi.org/10.1109/HPCA51647.2021.00038

[2] H. Lipmaa, P. Rogaway, and D. Wagner, "Ctr-mode encryption, comments to nist concerning aes modes of operations," in *NIST Workshop on Modes of Operation*, 2000.

[3] G. Saileshwar, P. Nair, P. Ramrakhyani, W. Elsasser, J. Joao, and M. Qureshi, "Morphable counters: Enabling compact integrity trees for low-overhead secure memories," in *51st Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*. IEEE, 2018.

[4] G. Saileshwar, P. J. Nair, P. Ramrakhyani, W. Elsasser, and M. K. Qureshi, "Synergy: Rethinking secure-memory design for error-correcting memories," in *IEEE International Symposium on High Performance Computer Architecture (HPCA)*. IEEE, 2018.

[5] K. A. Zubair and A. Awad, "Anubis: ultra-low overhead and recovery time for secure non-volatile memories," in *Proceedings of the 46th International Symposium on Computer Architecture*. ACM, 2019.