ExtraCC: Improving Performance of Secure NVM with Extra Counters and ECC

Zhengguo Chen*, Youtao Zhang† and Nong Xiao‡
*School of Computer, National University of Defense Technology, Changsha, China
†Department of Computer Science, University of Pittsburgh, Pittsburgh, USA
‡School of Data and Computer Science, Sun Yat-sen University, Guangzhou, China
Email: zgchen.nudt@foxmail.com, zhangyt@cs.pitt.edu, xiaon6@mail.sysu.edu.cn

***Abstract*—Emerging non-volatile memories (NVMs), while exhibiting great potential to be DRAM alternatives, are vulnerable to security attacks. Adopting counter mode AES based encryption and authentication schemes help to protect memory security but tend to incur non-negligible performance overhead in order to keep data consistency between counters and user data. In particular, counters are associated with logical addresses such that counters of hot data may overflow frequently, incurring lifetime and performance overhead in secure NVM system. The recently proposed ACME scheme mitigates the issue by associating counters with physical addresses and leveraging underlying wear-leveling schemes. While it stores and updates data and counters together, it destroys counter locality and introduces large read overhead during integrity check.**

In this paper, we propose ExtraCC to address the performance and lifetime losses in secure NVMs. We keep an extra counter and enhance the ExtraCC with logical-addressed-physical-associated (LAPA) counter scheme and two-tiered ECC to not only keep the counter locality but also effectively reduce write overhead. Our experimental results show that it achieves 15.2% performance improvement and 20.5% write traffic reduction over the state-of-the-art, with about 8.4% storage overhead.

***Index Terms*—non-volatile memories, security, ECC**

I. Introduction

Emerging non-volatile memories (NVMs) have high potential to be alternatives to DRAM due to low power, high density and scalability, e.g., ReRAM, PCM and STT-RAM [15], [16]. However, modern computer systems that integrate NVMs are vulnerable to classic security attacks, including confidentiality and integrity attacks [14]. The former includes bus snooping and memory scanning attacks [3], [23] while the latter includes splicing, spoofing and replay attacks [2], [9]. A traditional secure memory system often adopts Galois Counter Mode (GCM) [7] based encryption and authentication schemes [17], which use counter mode AES algorithm to encrypt plaintext, and builds a Merkle tree (MT) [10] on MACs (message authentication code) to protect data integrity. To mitigate the storage overhead on storing the Merkle tree, Yitbarek *et al.* proposed to exploit MAC as ECC (error correcting codes) [20], which effectively reduces storage overhead at the cost of limiting the error correction capability. Costan *et al.* proposed Bonsai Merkle Tree (BMT) [10] that builds the merkle tree on counters instead of MACs to reduce MT overhead.

It is challenging to secure NVMs — a secure NVM memory system stores security data in four areas that hold ciphertext, counters, MACs, and MT nodes, respectively. To keep data consistency, updates to the memory lead to multiple write operations to the NVM devices, which not only degrades the system performance but also shortens the NVM chip lifetime. Conventional GCM encryption associates counters with logical addresses such that the counters of hot data may overflow frequently [12]. To address this issue, Swami *et al.* proposed ACME [12] to associate counters with physical addresses and leverage underlying wear-leveling schemes. The write performance was improved as counters are saved along with ciphertext, which reduces the number of NVM writes at runtime and effectively improves the write performance.

However, ACME complicates the hardware address mapping as well as the data transfers, in particular, the data to be written are transferred in 9 beats instead of 8 beats in DRAM standards. ACME focuses on solving counter overflow and improving write performance [12], which destroys the counter locality and slows down the read performance. For a system that enforces both confidentiality and integrity protection, the system needs to load consecutive counters to compute a checking hash. The non-consecutive storing of counters tends to introduce more read operations and degrades the system performance. On the other hand, Osiris [19] exploits ECC to mitigate counter update overhead to reduce write overhead. However, it utilizes split-counter scheme [17] and logical-address associated counters to keeps counter locality. Thus it still suffers from counter overflow [12]. In conclusion, existing designs cannot solve counter overflow issue while keeping high performance and low write overhead.

In this paper, we first identify the challenges in secure NVM — while ACME mitigates the counter overflow issue, it breaks the counter locality and thus increases counter access overhead. We then propose ExtraCC that saves two counters to solve counter overflow issue and speed up both read and write operations. For the two counters, one is stored together with other counters for consecutive memory lines while the other occupies a part of the ECC of each memory line to improve write performance. We utilize LAPA counter scheme to solve counter overflow issue meanwhile keep counter locality. And we adopt a two-tiered ECC that splits the error detection and error correction capability so that saving extra counters does not complicate either the address mapping or the data transfers. We evaluate ExtraCC and compare it with the state-of-the-art.

Our experimental results show that ExtraCC achieves 15.2% performance improvement and 20.5% write traffic reduction over the state-of-the-art, with about 8.4% storage overhead.

II. Background and Motivation

A. Threat Model

When designing a secure system, it is a common practice to assume a trusted computing base (TCB) and define a set of valid threats that constitute the threat model of the secure system, as shown in Figure 1. Under the traditional trusted computing setting, the TCB includes the processor only, i.e., off-chip components, such as memory buses and memory modules are vulnerable to security attacks. Core parts of the OS are also secure.

Similar as the secure memory designs in the literature [10], [17], we are to enforce three types of protections: (1) we are to encrypt the user data and code to protection data confidentiality; (2) we are to authenticate the user data to prevent attackers compromising the data; (3) we are to check the overall data integrity to prevent data splicing, spoofing, and replay attacks.

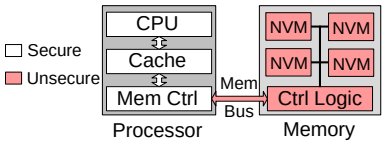

Fig. 1. Threat Model.

B. GCM based Encryption and Authentication

Most encryption and authentication schemes adopt Galois Counter Mode (GCM) [7] to protect memory security of traditional DRAM memory. Figure 2 shows the procedure of counter mode encryption and message authentication code (MAC). The encryption is at cache line granularity, i.e., 64B. To encrypt, the system generates a one-time pad (OTP) by applying AES encryption on a seed that includes the data address, a counter value (Ctr), and a constant encryption initialization vector (EIV). The system then generates the ciphertext (plaintext) by XORing the OTP with the plaintext (ciphertext) for encryption (decryption). Same OTP will incur security risk of information leakage [10], [12], [17]. Accordingly, a limited-size counter, e.g., 32 bits, is attached to each memory line so that each line update increments the counter, which avoids generating the same OTP for encryption/decryption. When any counter overflows, the system needs to change the EIV and re-encrypt all cache lines, which incurs large overhead. To reduce storage and overflow overhead of counters, *split-counter* scheme [17] encodes counter as a concatenation of a large major counter and a small minor counter, as shown in Figure 3(a). Then a counter cacheline packs a 64-bit major counter and 64 7-bit minor counters. When a minor counter overflows, it will increment the major counter, reset all 64 minor counters, and update corresponding 64 ciphertexts as well as their metadata.

To prevent data replay attacks, the system generates one MAC for each memory line using GHASH [17], builds a MT

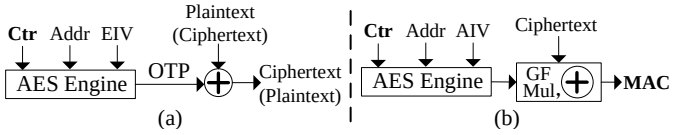

Fig. 2. Procedure of encryption and decryption (a) and MAC (b).

on MACs, and checks the secure root before each memory read access [2], [9]. As shown in Figure 2(b), the MAC is generated from the ciphertext together with the data address, the counter and an initialized vector (AIV). A MAC is usually 128 bit long. The root of the Merkle tree is stored on chip that is secure. Given one memory line can store four MACs, enforcing memory authentication tends to introduce significant storage overhead. As shown in Figure 3(b), BMT proposed to build the Merkle tree on counters rather than MACs by iteratively computing a 8B hash for 64B data until to the BMT root. a 64B memory line can store 64 counters with *split-counter* scheme [19], which reduces the height of the tree. In addition to ciphertext, the security metadata, i.e., counters, MT nodes, and MACs, are also stored in memory.

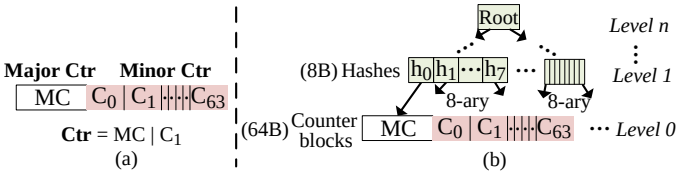

Fig. 3. Procedure of split-counter scheme (a) and BMT (b).

C. Secure NVMs

Secure NVMs, while we may adopt existing secure memory designs, faces new challenges. In particular, NVM often needs to enforce data persistence so that a power failure shall not leave the system in inconsistent state. For this reason, updating user data results in updating both the ciphertext and the security metadata to NVM at the same time, which converts a user write request to four NVM writes that update ciphertext, counter, MAC, and MT node, respectively.

Figure 4(a) shows the basic secure NVM structure with ciphertext and metadata saved in four different regions [19]. Normally, we cache metadata on-chip to avoid extra memory access [19]. A read memory access needs to (1) read the counter to generate the OTP; (2) read the ciphertext to decrypt; (3) read the MAC to ensure the ciphertext is not compromised; (4) read the MT to regenerate the secure root to prevent replay attack. All metadata read will access cache first and only access NVM on cache miss. In this paper, we restrict to BMT that combines split-counter scheme. Given the counter are stored together, reading one counter brings in the counters of neighboring memory lines. While the decryption just needs one counter to generate the OTP, building the BMT uses the whole line. This is beneficial as keeping more counters in the leaf node of the MT results in a small tree. A write memory access needs first decrypts the ciphertext as read operation. Then considering persistence, it writes updated ciphertext and

metadata in both cache and NVM. We assume the secure root is kept in a secure non-volatile register [19]. STASH [14] proposed to avoid writing through MT nodes to NVMs during data update. Given the secure root is saved, the tree can be rebuilt with up-to-date counters. Thus, by updating the MT nodes in cache during data update, a write request results in three NVM writes, e.g., ciphertext, counter, MAC.

Conventional GCM encryption associates counters with logical addresses. However, counters associated with hot data tend to overflow frequently [12], incurring non-negligible overhead. To address this issue, ACME proposed to leverage underlying Start-Gap wear-leveling scheme to realize counter write leveling, which associates counters with physical addresses [12]. As shown in Figure 4(b), ACME stores counters along with their respective data (ciphertext), where Ctr_3 and Ctr_4 are associated to physical addresses 3 and 4, respectively. When the gap line exchanges with the logical ciphertext D due to wear-leveling scheme, Ctr_3 and Ctr_4 keep unchanged. Accordingly, the D uses Ctr_4 to produce OTP in the future instead of Ctr_3. In this way, all the counters can 'wear' evenly after wear-leveling. Thus, they can use short-sized counters, e.g., 24 bits.

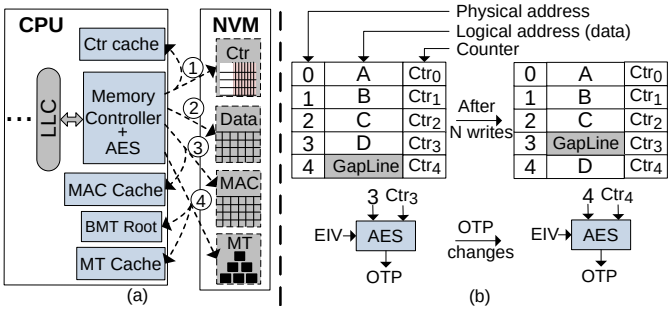

Fig. 4. Basic secure NVM architecture (a) and ACME (b).

D. Motivation

In this paper, we target at providing a full fledged secure NVM system. The baseline adopts GCM based encryption and authentication and builds BMT enhanced by split-counter scheme. However, a secure NVM is both memory read and write intensive, making it challenging to speed up with existing schemes. For example, ACME [12] saves counters together with their ciphertext, shown in Figure 4. However, it destroys counter locality. It becomes difficult to build a BMT with the new data allocation — given a 32-bit counter, a naive implementation would need to read 16 memory lines to collect 16 counters. In addition, saving 512b ciphertext and 32-bit counter together complicates the address mapping as well as the memory control logic. Due to low counter locality, ACME reads cannot full exploit counter cache so that the read performance is usually lower than the baseline. We compare the number of counter read from NVM when adopting ACME in secure NVM system. The numbers are normalized to the baseline. As shown in Figure 5, ACME increases counter reads by 74.4× on average. This is because baseline keeps counter locality with split-counter scheme and lots of counter reads hit in cache.

Another issue is counter update. For each memory write request, ACME writes counter and ciphertext together in 9-beats instead of 8 beats in DRAM standards, which needs hardware support. Osiris utilizes ECC to recover counters and only updates counters in cache. However, it still needs to write counter to NVM in every 4 updates due to recovery overhead [19]. And it suffers from counter overflow because it associates counters with logical addresses [12]. To summarize, it remains a challenging task to speed up a full fledged secure NVM that enforces encryption, authentication, and integrity protection.

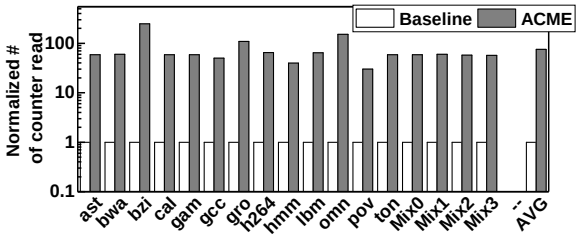

Fig. 5. Normalized number of counter read from NVM.

III. ExtraCC Design

In this section, we present the detailed design of ExtraCC. We elaborate the different access scheme and discuss the system security and availability.

A. Overview of ExtraCC

Figure 6(a) presents the overview of ExtraCC scheme. Each memory line in ExtraCC saves two 32-bit counters: a permanent counter is saved in a reserved area together with counters of the neighboring memory line; and a working counter is saved together with the memory line but occupies a portion of the ECC space. The permanent counters are similar to those in the baseline and Osiris that keep counter locality while the working counters are similar to those in ACME. Accordingly, we adopt a two-tier ECC design that splits the ECC of data to two parts. A 16-bit T1EC that is responsible for error detection and a 64-bit T2EC that is responsible for error correction. Also, we propose LAPA counter scheme that uses logical address to locate counter but the counter is associated with physical address of data (ciphertext). This scheme combines ACME principle to solve counter overflow and meanwhile keeps counter locality. Same to baseline, the last level cache (LLC) and relative metadata caches hold the plaintext, preloaded counters, MACs and MT nodes.

We next briefly describe how to perform secure NVM read and write requests. As described in section II-C, conventional secure scheme needs read 4 kinds of security metadata for serving a read request while incurs three instant writes to NVM and one write back of MT nodes for serving a write request. Figure 6 highlights special counter access scheme in ExtraCC. To read a memory line from the secure NVM, ExtraCC reads ciphertext as well as the metadata. Note that it gets two counters belong to the requested ciphertext: one from working counter and the other one from the permanent counter region. Actually, ExtraCC only read permanent counter block (64B) instead of working counters into counter cache. As

shown in Step 1 of Figure 6, it gets all needed counters for decryption/integrity check by reading the counter block (64B) in the permanent counter area. Again, it reads the counter cache line first. If the needed counter hits in the cache, it overlaps the OTP generation with the reading of other data. In ExtraCC, it stores complete 32-bit counters instead of using split-counter scheme. Thus each counter line in permanent counter area has 16 counters.

For a memory write, ExtraCC need finish read operation first to decrypt accessed ciphertext, e.g., D_0 in Figure 6(a). Then it updates D_0 and all its metadata. Here, ExtraCC updates both of the two counters, i.e., the working counter (Ctr_0) in ECC area and the permanent counter of D_0 in counter cache, as shown in Step 2. Updated counter cache line will be marked as dirty and write to permanent counter area in NVM during eviction. Then it updates D_0 and MAC in NVM, and updates MT nodes in MT cache as well as BMT root. ExtraCC need not update permanent counter in NVM instantly, so that it will not incur extra NVM write. In this way, both of the counter cache and the working counter keep the latest counter. When ExtraCC reads a counter cache line for serving a read request, if it hits the cache, it can read the latest counter cache line. Otherwise, it has been evicted and the permanent counter area in NVM keeps the latest version. With the existing ECC bus, for write operation, ExtraCC incurs two instant writes to NVM, i.e., one for D_0+Ctr_0 and one for MAC, and results in two write back updates of counter cache line and MT nodes. Note that system crash may lead to inconsistency of working counters and permanent counters. However, ExtraCC can go through the working counters to recover permanent counters.In the case of power failure, if data in counter area and MT nodes area are not updated in time, it will not influence the system availability, which is discussed in subsection III-D.

In conclusion, ExtraCC combines the advantages of conventional secure scheme and ACME. It keeps the counter locality by reading the counter cache line and gets all needed counters for integrity check in one memory read. Also it reduces instant memory writes by writing ciphertext and respective counter together with existing ECC bus. Then we propose two-tiered ECC to arrange existing ECC and LAPA counter scheme to address counter overflow issue.

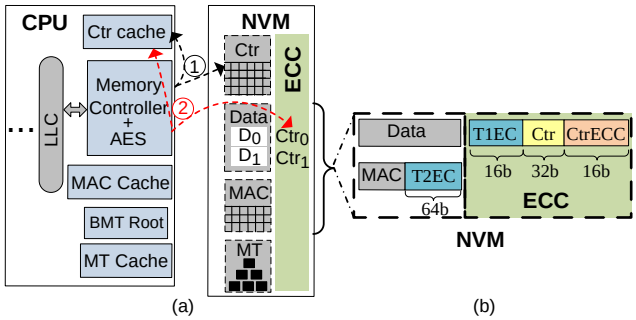

Fig. 6. Overview of ExtraCC (a) and two-tiered ECC (b).

B. Two-tiered ECC

A traditional ECC chip provides SEC-DED (single error correction, double error detection) capability through parity codes, e.g., Hamming and Hsiao [21]. In this paper, we adopt a two-tiered ECC design [21], [22] to enable saving counters in the ECC chip without changing the memory architecture.

Generally, it uses 64-bit ECC for a 512-bit ciphertext. Yoon *et al.* [22] proposes two-tiered ECC and constructs a T1EC for detection and a T2EC for correction. Assigning a 16-bit T1EC for a single cache line can detect at most 31-bit errors with no correction ability [22]. Thus, ExtraCC adopts the 16-bit T1EC to detect errors in ciphertext while adopts 64-bit conventional ECC as T2EC to provide SEC-DED. Figure 6(b) presents the detail of two-tiered ECC. The T1EC and a 32-bit counter are stored in original ECC area while T2EC is stored alongside respective MAC. In other words, we move the original ciphertext ECC and make room for extra counter and T1EC. Note that the T1EC only detect errors in respective ciphertext and itself without the extra counter. Thus, the remained 16 bits can construct a small counter ECC (CtrECC) to protect the extra counter and provide SEC-DED.

When serving a write request, T1EC, T2EC and the CtrEcc will be updated instantly. In this scenario, T1EC, counter and CtrECC can be written together with ciphertext to NVM while T2EC can be written together with MAC in one NVM write. Thus, two-tiered ECC will not incur additional NVM writes. However, when serving a read request, only T1EC are read with respective ciphertext and detect whether errors occur in the ciphertext. Only when T1EC detects some errors, we read T2EC to correct. Subsection III-D will discuss how the proposed two-tiered ECC collaborates with MAC to protect the system availability.

C. LAPA Counter Scheme

Memory accesses often exhibit significant imbalance, in particular, a few logical addresses are likely to be accessed extremely frequently. Given the schemes that associate counters with logical addresses are likely to have counter overflows, we associate the counters with the physical addresses of data. However, two counters in ExtraCC makes it complicate. Referring to ACME scheme [12], we propose LAPA scheme to solve counter overflow issue.

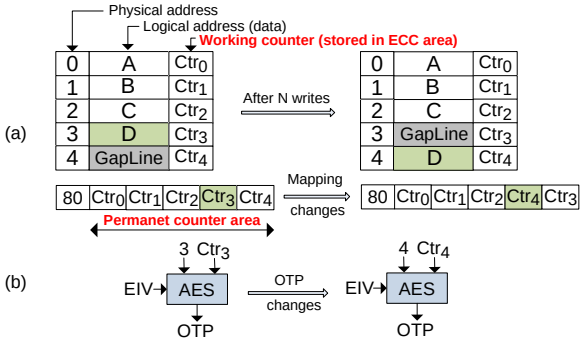

Fig. 7. LAPA counter scheme during wear leveling.

Figure 7 elaborates the LAPA scheme. we store the counters from consecutive physical memory lines together in permanent counter area while we associate a working counter in the freed ECC space from the two-tiered ECC design. All working counters are associated with physical addresses. Permanent

counter area is addressed by logical address, e.g., initially the first four consecutive counters $Ctr_0 \sim Ctr_3$ refer to data $A \sim D$ whose logical addresses are consecutive. Then The Start-Gap wear-leveling scheme [8] exchanges the Gapline with D. Same to principle of ACME, D will use Ctr_4 as its counter in future to realize counter write leveling and solve counter overflow (shown in Figure 7(b)). However, we exchange Ctr_3 and Ctr_4 in permanent counter area, since logically the fourth counter in this area is belong to D (shown in Figure 7(a)). Thus, after wear leveling, the permanent counters still keep locality meanwhile we solve counter overflow.

When reading a memory line, we read the counter either from the copy in the counter cache, or the permanent counter area in NVM. When writing a memory line, we update the working counter in NVM and the permanent counter copy in the counter cache. The permanent counter keeps the stale copy in NVM until the cached copy is flushed. If a power failure leads to loss of the cached copy, after system reboots, the system will go through all memory lines to check if the working counter is the same as the permanent counter.

D. System Security and Availability

Compared to conventional secure NVM schemes, ExtraCC uses two counters and two-tiered ECC. Given ExtraCC does not change the GCM encryption and authentication process and the BMT structure, it maintains the same security level as that in the baseline. If a power failure leaves the two counters in inconsistent state, ExtraCC ensures that the working counter is always up-to-date so that there is no ambiguity.

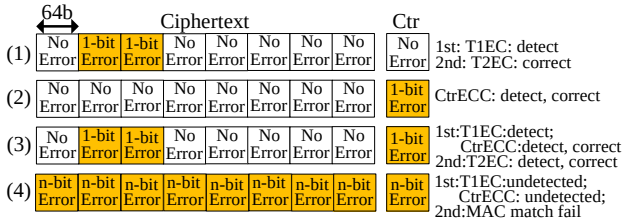

Fig. 8. Potential errors.

We next elaborate how the two-tiered ECC collaborate with MAC to keep system secure and available. We discuss the two-tiered ECC scheme in two situations: normal access and security attack. In the case of normal access, we only consider at most 1-bit error every 64 bits. Otherwise, it will exceed the correction ability of normal ECC. Figure 8 shows four kinds of errors. In the first case, if it occurs at most one-bit error in every 64-bit data block, T1EC can detect the error. Then it will read T2EC to correct the errors. In the second case, an error occurs in the extra counter, the CtrECC can detect and correct the error. In the third case, if both of the ciphertext and the extra counter occurs errors, T1EC detects errors in ciphertext and CtrECC detects and corrects the error in counter. Then it also reads T2EC to correct errors in ciphertext.

In the situation of security attack, both of T1EC and CtrECC fail to detect and correct errors. Thus, the ciphertext will be considerred as correct. However, failure in MAC match and following integrity check will distinguish the security attack. In conclusion, extra counter and two-tiered ECC will not influence the system security and availability.

IV. Experiment

In this section, we present experimental results under various workloads and evaluate the effectiveness of ExtraCC.

A. Experimental Methodology

To evaluate the effectiveness of our scheme, we compare our scheme to the stat-of-the-art using a trace-driven in-house simulator. We use the PIN tool [1] to collect traces of SPEC CPU2006. We run all the benchmarks for 400 millions instructions after skipping the warm up phase.

Table I shows the details of the settings. The simulator models a full memory hierarchy, including 2-level caches with LRU cache replacement scheme. We select PCM as an example of NVMs. We use NVSim [4] to simulate energy consumption of each read and write operation. We also mix some workloads for better evaluations. And we set the OTP generation latency of 128-bit AES engine as 72ns [11], [12]. In this paper, we compare the following four schemes:

—Baseline. It uses conventional store organization that stores data, counters, MACs and MT nodes in specific area. It utilizes GCM and BMT to protect confidentiality and integrity.

—ACME. It stores counters alongside their respective ciphertext as a single memory block. During a data transfer, it uses nine data beats to read and write ciphertext and respective counter together.

—Osiris. It enhances the baseline with Osiris scheme [19] that utilizes ECC to mitigate counter update overhead. During memory write, it only updates counters in cache and writes to NVM in every 4 updates to reduce recovery overhead [19].

—ExtraCC. It deploys extra counters and ECC, and follows the store scheme described in this paper.

Both of Baseline and Osiris utilize *split-counter* [17] scheme.

TABLE I
Configurations [12], [15], [19] and mixed workloads

CPU	4 cores single issue in-order CMP, 4GHz
L1 I/D-cache	Private, 32KB, 2-way, 64B block, 2 cycles
L2 cache	Shared, 2MB, 4-way, 64B block, 10 cycles
Counter Cache	Shared, 128KB, 4-way, 64B block, 2 cycles
MAC Cache	Shared, 32KB, 4-way, 64B block, 2 cycles
MT Cache	Shared, 16KB, 4-way, 64B block, 2 cycles
PCM memory	8GB, 1 channel, 2 ranks, 8 banks/rank, 8-entry write queue/bank, Latency: Read: 100ns, Set: 200ns , Reset: 100ns, energy: Read: 1.49 nJ, Set:6.76 nJ, Reset: 6.73 nJ
Workload	Benchmarks
Mix0	astar, bwaves, bzip2, calculix
Mix1	bwaves, calculix, gcc, gamess
Mix2	lbm, h264, tonto, gromacs
Mix3	hmmer, lbm, omnetpp, povray

B. Storage Overhead and Recovery Time

ExtraCC removes original ECC for ciphertext to MAC area as T2EC and introduces additional storage overhead of extra counters and T1EC. It adds additional 64 bits for each 512-bit ciphertext. Taking 4GB ciphertext for example, it needs 1GB

MAC, 256MB counters, approximate 22MB MT nodes and 675MB ECC for security metadata. ExtraCC involves 512MB additional storage overhead of extra counters and T1EC, which incurs about 8.4% storage overhead. During recovery after system crashes, ExtraCC will go through the working counters to recover counters in permanent counter area. For 8GB NVM in this paper, it takes only 14.3s, close to 13.7s of Osiris.

C. Performance

We first compare the performance of all schemes. Figure 9 shows the normalized execution time. Compared to Baseline, ACME and Osiris improve the performance by 12.8% and 13.9% on average, respectively. ACME solves the counter overflow issue while it breaks the counter locality. On the contrary, Osiris keeps the counter locality while it suffers from counter overflow. However, ExtraCC realizes averagely 16.1% and 15.2% performance improvement than ACME and Osiris. This is because ExtraCC solves counter overflow meanwhile keeps counter locality with LAPA counter scheme. Also, ExtraCC reduces NVM write of counter update and further improves performance, which will be discussed next.

D. Read Response Time

Figure 10 compares the normalized read response time in different schemes. Both of Osiris and ExtraCC are very close to baseline. However, ACME increases the read response time by 49.9%. Counter locality affects the counter cache hit rate during decryption and integrity check, which plays a key role in read response time. Both of Osiris and baseline utilize split-counter scheme to keep counter locality while ACME stores counters along with ciphertext, breaking the counter locality totally. However, ExtraCC reads counters in counter area to keep the spatial locality. It fetches multiple counters in one read, which significantly reduces memory reads.

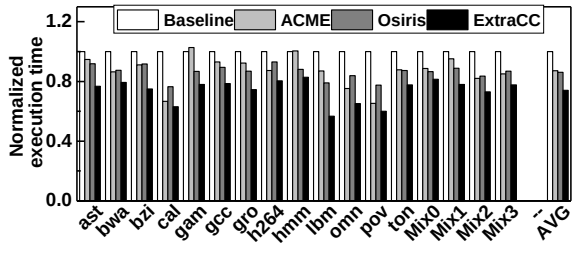

Fig. 9. Normalized execution time in various schemes.

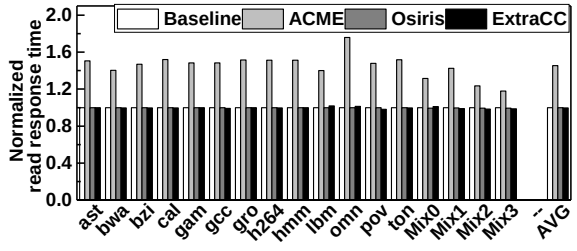

Fig. 10. Normalized read response time in various schemes.

E. Write Traffic

Figure 11 shows the normalized number of NVM writes in various schemes. Compared to Baseline, averagely ACME and ExtraCC reduce the NVM writes by 40.7% and 40.6%, respectively. This is because both of them update counter and ciphertext in one NVM write during memory write (ACME stores counter alongside ciphertext and ExtraCC writes working counter with ECC bus). However, ExtraCC still have to write dirty counter cache line to permanent counter area during eviction, which introduces slightly more NVM writes than ACME. Compared to Osiris, ExtraCC achieves 20.5% NVM write reduction. This is mainly because Osiris suffers from counter overflow that incurs extra NVM writes. Also, Osiris updates dirty counter cache lines to NVM periodically [19]. Thus, ExtraCC improves the system lifetime effectively.

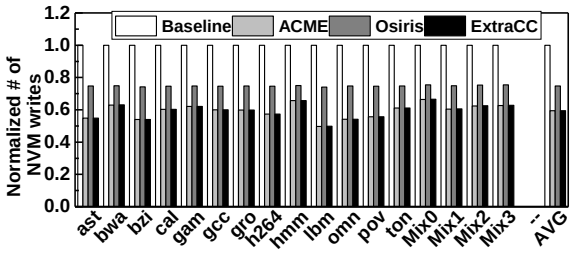

Fig. 11. Normalized number of NVM writes in various schemes.

F. Energy Consumption

Figure 12 shows the energy consumption of NVM in different schemes. In our experiment, we observe that NVM write operation is much more than NVM read. This is because many read operations hit the on-chip caches. However, many writes including ciphertext and metadata updates are issued to NVM due to persistence. Thus, energy consumption mainly depends on write traffic, which has similar tendency with Figure 11. To summarize, ExtraCC reduces up to 31.9% (lbm) and 1.5% (omn) of energy consumption over the Osiris and ACME.

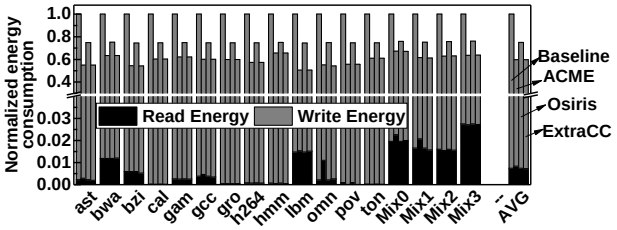

Fig. 12. Normalized read and write energy consumption of different schemes.

V. Related Work

Many related solutions have been proposed for secure NVMs [6]. DEUCE [23] and SECRET [13] proposed to improve encryption according to data content. SuperMem [25] and [18] utilized persistent domain in memory controller to reduce counter update overhead. These techniques are excellent to reduce encryption and counter update penalty, but they only consider confidentiality attacks. ExtraCC focuses on confidentiality attacks and integrity attacks together and it is

compatible to these techniques. Zubair *et al.* proposed Anubis to reduce recovery time of TB-level NVM [24]. Liu *et al.* [5] proposed to relax persisting counters for non-persistent data and the effectiveness depends on ratio of non-persistent data [19].

VI. Conclusion

In this paper, we proposed ExtraCC to address the challenges in designing a secure NVM system. ExtraCC uses two counters to speed up both read and write operations when performing security checks. Our experimental results show that ExtraCC achieves 15.2% performance improvement and 20.5% write traffic reduction over the state-of-the-art (Osiris), with about 8.4% storage overhead. Also, it improves the performance by 16.1% than ACME.

Acknowledgment

This work is supported by the National Key Research and Development Program of China under Grant NO. 2019YFB1804502, the National Natural Science Foundation of China under Grant NO. 61832020, 61802418, the Natural Science Foundation of Guangdong Province under Grant No. 2018B030312002, the Major Program of Guangdong Basic and Applied Research under Grant No. 2019B030302002, the Program for Guangdong Introducing Innovative and Entrepreneurial Teams under Grant NO. 2016ZT06D211, Key-Area Research and Development Program of Guangdong Province under Grant No. 2019B010107001. The authors thank the anonymous reviewers for their constructive comments.

References

[1] "Pin tool," https://software.intel.com/en-us/articles/pintool.

[2] S. Aga and S. Narayanasamy, "Invisimem: Smart memory defenses for memory bus side channel," *ACM SIGARCH Computer Architecture News*, vol. 45, no. 2, pp. 94–106, 2017.

[3] A. Awad, Y. Wang, D. Shands, and Y. Solihin, "Obfusmem: A low-overhead access obfuscation for trusted memories," in *Proceedings of the 44th Annual International Symposium on Computer Architecture*, 2017, pp. 107–119.

[4] Q. Hu, G. Sun, J. Shu, and C. Zhang, "Exploring main memory design based on racetrack memory technology," in *Proceedings of the 26th edition on Great Lakes Symposium on VLSI*, 2016, pp. 397–402.

[5] S. Liu, A. Kolli, J. Ren, and S. Khan, "Crash consistency in encrypted non-volatile main memory systems," in *2018 IEEE International Symposium on High Performance Computer Architecture (HPCA)*. IEEE, 2018, pp. 310–323.

[6] S. Liu, K. Seemakhupt, G. Pekhimenko, A. Kolli, and S. Khan, "Janus: Optimizing memory and storage support for non-volatile memory systems," in *2019 ACM/IEEE 46th Annual International Symposium on Computer Architecture (ISCA)*. IEEE, 2019, pp. 143–156.

[7] D. McGrew and J. Viega, "The galois/counter mode of operation (gcm)," *submission to NIST Modes of Operation Process*, vol. 20, p. 10, 2004.

[8] M. K. Qureshi, J. Karidis, M. Franceschini, V. Srinivasan, L. Lastras, and B. Abali, "Enhancing lifetime and security of pcm-based main memory with start-gap wear leveling," in *2009 42nd Annual IEEE/ACM international symposium on microarchitecture (MICRO)*. IEEE, 2009, pp. 14–23.

[9] J. Rakshit and K. Mohanram, "Assure: Authentication scheme for secure energy efficient non-volatile memories," in *Proceedings of the 54th Annual Design Automation Conference 2017*, 2017, pp. 1–6.

[10] B. Rogers, S. Chhabra, M. Prvulovic, and Y. Solihin, "Using address independent seed encryption and bonsai merkle trees to make secure processors os-and performance-friendly," in *40th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO 2007)*. IEEE, 2007, pp. 183–196.

[11] W. Shi, H.-h. S. Lee, M. Ghosh, C. Lu, and A. Boldyreva, "High efficiency counter mode security architecture via prediction and precomputation," in *32nd International Symposium on Computer Architecture (ISCA'05)*. IEEE, 2005, pp. 14–24.

[12] S. Swami and K. Mohanram, "Acme: Advanced counter mode encryption for secure non-volatile memories," in *2018 55th ACM/ESDA/IEEE Design Automation Conference (DAC)*. IEEE, 2018, pp. 1–6.

[13] S. Swami, J. Rakshit, and K. Mohanram, "Secret: Smartly encrypted energy efficient non-volatile memories," in *Proceedings of the 53rd Annual Design Automation Conference*, 2016, pp. 1–6.

[14] S. Swami, J. Rakshit, and K. Mohanram, "Stash: Security architecture for smart hybrid memories," in *2018 55th ACM/ESDA/IEEE Design Automation Conference (DAC)*. IEEE, 2018, pp. 1–6.

[15] R. Wang, L. Jiang, Y. Zhang, and J. Yang, "Sd-pcm: Constructing reliable super dense phase change memory under write disturbance," *ACM SIGARCH Computer Architecture News*, vol. 43, no. 1, pp. 19–31, 2015.

[16] C. Xu, D. Niu, N. Muralimanohar, N. P. Jouppi, and Y. Xie, "Understanding the trade-offs in multi-level cell reram memory design," in *2013 50th ACM/EDAC/IEEE Design Automation Conference (DAC)*. IEEE, 2013, pp. 1–6.

[17] C. Yan, D. Englender, M. Prvulovic, B. Rogers, and Y. Solihin, "Improving cost, performance, and security of memory encryption and authentication," *ACM SIGARCH Computer Architecture News*, vol. 34, no. 2, pp. 179–190, 2006.

[18] F. Yang, Y. Lu, Y. Chen, H. Mao, and J. Shu, "No compromises: Secure nvm with crash consistency, write-efficiency and high-performance," in *2019 56th ACM/IEEE Design Automation Conference (DAC)*. IEEE, 2019, pp. 1–6.

[19] M. Ye, C. Hughes, and A. Awad, "Osiris: A low-cost mechanism to enable restoration of secure non-volatile memories." Sandia National Lab.(SNL-NM), Albuquerque, NM (United States), Tech. Rep., 2018.

[20] S. F. Yitbarek and T. Austin, "Reducing the overhead of authenticated memory encryption using delta encoding and ecc memory," in *2018 55th ACM/ESDA/IEEE Design Automation Conference (DAC)*. IEEE, 2018, pp. 1–6.

[21] D. H. Yoon and M. Erez, "Flexible cache error protection using an ecc fifo," in *Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis*. IEEE, 2009, pp. 1–12.

[22] D. H. Yoon and M. Erez, "Memory mapped ecc: low-cost error protection for last level caches," in *Proceedings of the 36th annual international symposium on Computer architecture*, 2009, pp. 116–127.

[23] V. Young, P. J. Nair, and M. K. Qureshi, "Deuce: Write-efficient encryption for non-volatile memories," *ACM SIGARCH Computer Architecture News*, vol. 43, no. 1, pp. 33–44, 2015.

[24] 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*, 2019, pp. 157–168.

[25] P. Zuo, Y. Hua, and Y. Xie, "Supermem: Enabling application-transparent secure persistent memory with low overheads," in *Proceedings of the 52nd Annual IEEE/ACM International Symposium on Microarchitecture*, 2019, pp. 479–492.