ASSURE: Authentication Scheme for SecURE Energy Efficient Non-Volatile Memories

Joydeep Rakshit Kartik Mohanram

Department of Electrical and Computer Engineering, University of Pittsburgh, Pittsburgh
joydeep.rakshit@pitt.edu kartik.mohanram@gmail.com

ABSTRACT

Data tampering threatens data integrity in emerging non-volatile memories (NVMs). Whereas Merkle Tree (MT) memory authentication is effective in thwarting data tampering attacks, it drastically increases cell writes and memory accesses, adversely impacting NVM energy, lifetime, and system performance (instructions per cycle (IPC)). We propose ASSURE, a low overhead, high performance Authentication Scheme for SecURE energy efficient (ASSURE) NVMs. ASSURE synergistically integrates (i) smart message authentication codes (SMACs), which eliminate redundant cell writes by enabling MAC computation of only modified words on memory writes, with (ii) multi-root MTs (MMTs), which reduce MT reads/writes by constructing either high performance static MMTs (SMMTs) or low overhead dynamic MMTs (DMMTs) over frequently accessed memory regions. Our full-system simulations of the SPEC CPU2006 benchmarks on a triple-level cell (TLC) resistive RAM (RRAM) architecture show that on average, SMMT ASSURE (DMMT ASSURE) reduces NVM energy by 59% (55%), increases memory lifetime by $2.36\times$ ($2.11\times$), and improves IPC by 11% (10%), over state-of-the-art MT memory authentication.

CCS Concepts: **Security and privacy** → **Tamper-proof and tamper-resistant designs**

Keywords: **Non-volatile Memories, Authentication, Merkle Tree**

1. INTRODUCTION

Resistance-class non-volatile memories (NVMs) such as phase-change memory (PCM) and resistive RAM (RRAM) [1, 2] are potential DRAM replacement technologies because of their superior scalability, low power consumption, and high data density. Whereas data persistence is a desirable property of NVMs, it exposes data to confidentiality attacks, motivating NVM encryption [3–5].

Although NVM encryption ensures data confidentiality, it does not guarantee data integrity, a core component of the secure computation model [6–8]. Data integrity is the ability to detect adversarial tampering of (i) stored data or (ii) data transactions to/from memory. Ensuring data confidentiality alongside data integrity necessitates (i) memory encryption for data confidentiality followed by (ii) memory authentication for integrity of encrypted data [6, 7]. State-of-the-art memory authentication solutions maintain a logical data structure, Merkle Tree (MT), whose nodes are obtained by recursive computation of message authentication codes (MACs) over memory blocks, where MAC constitutes a cryptographic signature of the input data. In an MT, a parent node MAC ensures integrity of its child node MACs. MT memory authentication ensures the integrity of fetched (written) memory blocks by verifying (updating) its MAC lineage upto the MT root on the secure processor [6–8].

This research was supported by NSF Award CCF-1217738.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from Permissions@acm.org.

DAC '17, June 18–22, 2017, Austin, TX, USA

© 2017 ACM. ISBN 978-1-4503-4927-7/17/06...\$15.00

DOI: <http://dx.doi.org/10.1145/3061639.3062205>

Although MT memory authentication ensures data integrity, it invariably incurs high NVM energy and performance penalty due to increased cell writes and memory accesses. Our simulations of SPEC CPU2006 [9] workloads show that MT memory authentication increases cell writes (NVM energy) to $5.8\times$ ($5.3\times$) and degrades system IPC to $0.65\times$ in comparison to a nominal encrypted triple-level cell (TLC) RRAM architecture. This motivates the development of low penalty solutions for NVM authentication.

ASSURE is an Authentication Scheme for SecURE energy efficient (ASSURE) NVMs. ASSURE preserves the memory authentication properties of the underlying MT and is compatible with all encryption architectures. ASSURE synergistically integrates smart message authentication codes (SMACs) and multi-root MTs (MMTs) for low penalty memory authentication. SMAC leverages the observation that only the modified words are re-encrypted on consecutive write-backs to a memory location [3, 4]. SMAC partitions the MAC at word-level granularity and recomputes only those MAC words corresponding to the re-encrypted words during a memory write; this eliminates cell writes from the redundant MAC computation of unmodified words.

ASSURE complements SMACs with multi-root MTs (MMTs) to decrease the MT reads/writes. MMTs maintain multiple smaller static/dynamic MTs, collectively spanning the memory, with their roots on the secure processor. Static MMTs (SMMTs) partition the memory into memory block groups (MBGs) and statically maintain individual MTs for each MBG. Since each individual MT spans a smaller number of blocks, they have fewer levels, substantially reducing the number of MT levels read/updated for authentication. However, SMMTs incur significant processor-side on-chip storage for maintaining multiple MT roots. We propose dynamic MMTs (DMMTs) for low storage overhead MMT, which leverage the spatial and temporal locality of memory accesses in practical workloads to dynamically predict the frequently (infrequently) accessed hot (cold) MBG(s). Hot (cold) MBG(s) are assigned to the smaller hot (larger cold) MT. ASSURE constructs the MMTs with SMAC nodes instead of classical MAC nodes, extending the cell write reduction advantages of SMACs to MMT nodes.

ASSURE is evaluated on a TLC RRAM architecture for NVM energy, lifetime, and system IPC, and compared to state-of-the-art bonsai MT (BMT) [7] in the presence of dual counter encryption (DEUCE) [3] and Smartly EnCRypted energy Efficient NVMs (SECRET) [4]. NVMain [10] is used to estimate NVM energy on memory access traces of SPEC CPU2006 benchmarks [9] generated using Intel Pin toolset [11]. Our simulations show that on average, SMMT ASSURE (DMMT ASSURE) reduces NVM energy by 59% (55%) over BMT. Our lifetime evaluations using an in-house lifetime simulator show that on average, SMMT ASSURE (DMMT ASSURE) improves memory lifetime by $2.36\times$ ($2.11\times$) over BMT. Our full-system evaluations on MARSS [12] of composite SPEC CPU2006 workloads show that on average, SMMT ASSURE (DMMT ASSURE) improves system IPC by 11% (10%) over BMT.

This paper is organized as follows. Section 2 provides a brief background on memory encryption and authentication, and motivates ASSURE. Section 3 discusses the ASSURE architecture. Section 4 presents the evaluations, and Section 5 is our conclusion.

Figure 1: **State-of-the-art bonsai MT (BMT)**: BMT maintains an MT over the counters, and a single layer of HMACs over encrypted data. The data HMACs include the encrypted data with its line address and counter as input, protecting encrypted data from spoofing (data), splicing (address), and replay (counter) attacks.

2. BACKGROUND AND MOTIVATION

This section discusses state-of-the-art encryption and authentication solutions for NVM security, and motivates the impact of memory authentication on NVM energy, lifetime, and system IPC.

2.1 Threat model

An ideal secure computing platform requires three cornerstone properties: (i) confidentiality, (ii) integrity, and (iii) availability [13]. However, system design simplification and feasibility requires a specific threat model that differentiates the threats that the system protects against, and those not considered as part of the model [13]. In this work, we consider a threat model encompassing attacks on data confidentiality and data integrity; our trusted computing base (TCB) consists of the processor and core parts of the operating system (e.g., security kernels), whereas the off-chip memory and the processor-memory bus are untrusted [6–8]. Data confidentiality attacks aim to obtain secret data stored in memory or data being transferred to/from memory, motivating memory encryption. However, encryption does not protect against integrity attacks, where the adversary alters the data stored in or being transferred to/from memory. Data integrity attacks can be categorized into spoofing, splicing, and replay attacks [6, 7]. In spoofing attacks, the adversary replaces an existing valid memory block with fake data. In splicing attacks, the attacker swaps the memory content between two locations. Finally, in replay attacks, the content of a memory location is reverted back to an older value.

It is widely accepted that data integrity attacks can be thwarted by memory authentication, which verifies the integrity of all off-chip communications to/from the secure processor [6–8]. To prevent both integrity and confidentiality attacks, memory authentication must be deployed concurrent with memory encryption.

2.2 Encryption and authentication in NVMs

Memory encryption is achieved by applying a block cipher over plaintext data [3–6]. Prior works advocate the use of counter-mode encryption (CME) to offset the latency of encryption/decryption during memory write/read [3–6]. In CME, the data is XORed with a one-time pad (OTP) generated using a block cipher, which includes a secret key and a seed as inputs for encryption. The seed is composed of the memory block address and an associated counter that increments on each memory write. However, the diffusion property [14] of encryption drastically increases cell writes, which is especially undesirable in NVMs due to their high write energy/latency [3]. DEUCE [3] reduces the cell writes by re-encrypting only the modified words on memory writes; SECRET [4] further improves encryption in MLC/TLC NVMs by preventing zero-word re-encryption and XOR-based energy masking.

Memory authentication of an encrypted NVM system guarantees the integrity of both encrypted data and the counters of CME, since tampering of either results in the generation of invalid plaintext during decryption. Prior works have advocated the use of Merkle Tree (MT) memory authentication, which is proven to be secure against spoofing, splicing, and replay attacks [6–8]. State-of-the-art

Figure 2: **Comparison of average cell writes, NVM energy, and system IPC (norm. to baseline) of encrypted RRAM baseline with an encrypted RRAM deploying memory authentication**, Memory authentication increases the cell writes (NVM energy) to $5.8\times$ ($5.3\times$), and degrades system IPC to $0.65\times$ in comparison to the baseline.

memory authentication schemes use keyed hash message authentication codes (HMACs), which utilize a cryptographic hash function (e.g., SHA-1) and a secret key to generate a hash signature that includes the data block along with its corresponding counter and line address as inputs [7, 8]. MT memory authentication maintains a hierarchical tree structure of these HMACs, with the data and counter as its leaf nodes. In an MT, each parent node is an HMAC signature of its children node HMACs (data/counter in case of leaf nodes). The secret HMAC key and the MT root is stored on the secure, tamper-proof processor, preventing spoofing, splicing, or replay attacks. During reads, the memory block integrity is ascertained by verifying its HMAC lineage upto the MT root; in contrast, writes result in recomputation of the HMAC lineage upto the MT root to reflect the new data.

The state-of-the-art MT memory authentication architecture is the bonsai MT (BMT) [7], illustrated in Fig. 1. BMT leverages the CME architecture and maintains an MT over only the counters of a memory line rather than over both counters and data. BMT keeps a single level of HMAC over the data memory, where the data HMACs use the encrypted data block, address, and the counter as input. Although the encrypted data is not protected by an MT, it is immune to replay attacks because the data HMACs include BMT-protected counters as input. Since the counter memory is considerably smaller than the data memory, the BMT has significantly fewer levels than an MT over both data and counters, reducing the MT reads/writes and improving system IPC. Without loss of generality, we refer to BMT as MT for the rest of the paper.

2.3 Memory authentication overhead in NVMs

The NVM write energy (latency) is higher in comparison to the read energy (latency), and also higher in comparison to DRAM write/read energy (latency) [15, 16]; these differences are exacerbated in multi-/triple-level cell (MLC/TLC) NVMs [17]. HMACs demonstrate strong diffusion property, similar to data encryption [14], resulting in a high cell write rate, and render NVM write-reduction techniques like [18, 19] ineffective in practice. Hence, the data and counter HMACs in the BMT incur significant NVM energy/latency overhead and lifetime reduction. Further, additional memory accesses are introduced when the counter MT is read/updated on each counter read/write. These reads/writes integral to memory authentication stall critical data/counter reads or writes to the same memory bank, degrading system IPC. In Fig. 2, we use simulations of SPEC CPU2006 workloads to show that BMT authentication over a DEUCE-encrypted TLC RRAM increases the cell writes (NVM energy) to $5.8\times$ ($5.3\times$) and degrades system IPC to $0.65\times$ in comparison to an unauthenticated DEUCE-encrypted TLC RRAM.

In summary, whereas memory authentication is indispensable to a secure computing platform, it degrades NVM energy, latency, lifetime, and system IPC. By reducing authentication cell writes and memory accesses, our work ASSURE provides a low penalty memory authentication solution for secure NVM platforms.

Key acronyms – MAC: message authentication code; HMAC: hash MAC; SMAC: smart MAC; MT: Merkle Tree; BMT: bonsai MT; MMT: multi-root MT; S/DMMT: static/dynamic MMT; MBG: memory block group.

3. ASSURE

In this section, we describe ASSURE, a low overhead NVM authentication solution that deploys (i) smart MACs to reduce cell updates on authentication-related memory writes and (ii) multi-root MTs to reduce memory accesses for MT authentication. ASSURE preserves the security properties of classical MT authentication and is compatible with all NVM encryption solutions. Without loss of generality, we consider DEUCE [3] over SECRET [4] for NVM encryption in our discussions, because of its simpler architecture.

3.1 Smart message authentication codes

Without exception, hashed message authentication codes (HMACs) are the primary units of memory authentication [6–8]. However, HMACs incur increased (decreased) write energy (lifetime) owing to a high cell write rate (refer Sec. 2.3). In this section, we propose smart message authentication codes (SMACs) as a solution to realize security equivalent to HMACs with reduced (improved) write energy (lifetime) through decreased cell writes.

Observation: Memory authentication must be integrated with memory encryption for secure, tamper-resistant memory content, as discussed in Sec. 2.1. State-of-the-art NVM encryption [3,4] performs selective re-encryption of only the modified words to generate new ciphertext on each memory write. However, classical HMAC computation does not exploit this partial re-encryption, and requires HMAC recomputation of the entire encrypted cache line; this results in redundant HMAC recomputation of the unmodified words.

Smart MAC (SMAC) design: The core advantage of SMACs over classical HMACs is that SMACs perform selective HMAC recomputation of the encrypted data by leveraging the partial re-encryption property of the underlying NVM encryption architecture. DEUCE partitions the cache line into words of equal width, and re-encrypts only the modified words during a memory write. SMAC partitions the HMAC at word-level granularity and recomputes only those words corresponding to the re-encrypted words during a memory write; this eliminates cell writes due to the redundant HMAC computation of unmodified words.

To achieve selective HMAC computation of only the modified words, SMAC splits the original encrypted cache line into two decoupled intermediate messages (IMs) corresponding to the modified and unmodified words. The IMs have the same length and partition boundaries as the encrypted cache line. The first (second) IM, IM_1 (IM_2) is constructed from the modified (unmodified) words with the unmodified (modified) words zeroed out. IM_1 (IM_2) is then provided as input to a keyed cryptographic hash function, generating the intermediate HMACs, IH_1 (IH_2). Similar to IMs, the IH s are also partitioned at word-level granularity. The final HMAC (FH) is constructed with IH_1 (IH_2) words for the corresponding modified (unmodified) word positions. For example, if word k of an encrypted cache line is modified (unmodified), word k of FH is constituted by word k of IH_1 (IH_2).

During a read, the SMAC requires meta-data to identify the modified/unmodified word positions of the previous write to an address for valid FH reconstruction of the fetched encrypted data. Note that the underlying DEUCE architecture records modified bits (modbits) to track the modified/unmodified words. SMAC leverages the DEUCE modbits for tracking modified/unmodified words, incurring zero memory overhead (Note that ASSURE provisions independent modbits if implemented over SECRET [4], since SECRET does not provision modbits). Since valid decryption in DEUCE and FH reconstruction in SMAC depends on the critical modbits, we propose modbit integrity protection through modbit inclusion in the original input assignment to IM_1 . The modbits assigned to IM_2 are always zeroed out, since IM_2 represents the unmodified words

Figure 3: SMACs eliminate redundant cell writes for unmodified words (2, 3, and 4) during HMAC computation. Between writes 1 (W_1) and 2 (W_2), word 1 gets modified in the original encrypted cache line, altering the intermediate HMAC 1 (IH_1), resulting in modification of word 1 in the final HMAC (FH); however, IH_2 is unaltered due to unmodified words 2, 3, and 4, eliminating redundant cell writes for unmodified words 2, 3, and 4 of the FH.

of the cache line. Due to the strong diffusion property of HMAC algorithms, any change in modbits is reflected in subsequent alteration of the FH words corresponding to the modified word positions, enabling modbit integrity protection.

Figure 3 illustrates SMACs over a sequence of 2 consecutive writes. Without loss of generality, consider a 64-bit encrypted cache line with a word size of 16 bits (represented in hexadecimal), and one modbit per word. For write 1 (W_1), only word 1 is modified, setting the modbit for word 1. IM_1 is given by the modified word 1 and the modbits, with the unmodified words 2, 3, and 4 zeroed. IM_2 is given by the unmodified words 2, 3, and 4, with the modified word 1 and the modbits zeroed. The IH s are generated by treating the IM s as inputs for the cryptographic hash function, with the FH obtained by selecting word 1 from IH_1 and the rest from IH_2 . On write 2 (W_2), word 1 is modified again, subsequently altering IM_1 and IH_1 , resulting in a modification to word 1 of FH. However, words 2, 3, and 4 of the FH are unmodified at W_2 due to the unmodified words 2, 3, and 4 in the original encrypted cache line, decreasing (increasing) NVM write energy (lifetime).

3.2 Multi-root Merkle Trees

In MT authentication, a single MT spans the counter memory with a single root on the secure processor. However, MT authentication incurs a high penalty of additional memory reads (writes) to fetch and verify (update) the corresponding MT branch of a read (written) counter memory block, degrading NVM energy and system IPC. In this work, we propose multi-root MTs (MMTs) that maintain multiple smaller MTs having fewer levels (with multiple corresponding roots on the secure processor) as a novel alternative to the classical single-root MT. The multiple roots of the MMT collectively span the entire counter memory, with each MT assigned to a distinct memory block.

Although static MMTs achieve substantial MT read/write reduction, they incur high secure processor-side storage overhead of multiple roots. ASSURE leverages the spatial and temporal locality of memory accesses in practical workloads to realize a prediction architecture that dynamically identifies and maintains a smaller MT over the frequently accessed memory block, while spanning all other memory blocks with a larger MT. This reduces storage overhead to only 2 roots on the processor.

3.2.1 Static multi-root Merkle Trees

We begin by discussing a centralized static MMT (SMMT) architecture. In this approach, we statically assign MTs to groups of memory blocks and maintain their roots on the secure processor, effectively reducing MT traversal levels, thereby improving NVM energy and system IPC.

Figure 4: (a) Classical single-root MT, with the traversal path for verification of leaf L_1 highlighted. (b) An SMMT with 2 memory block groups covered by 2 smaller MTs with 2 independent MT roots on the secure processor. In SMMT, the traversal path for L_1 verification has 1 less MT level, due to the smaller individual MTs.

Observation: In MTs, the leaf nodes represent the integrity-preserved memory blocks. In SMMTs, we leverage the observation that a smaller MT that spans fewer leaf nodes (memory blocks) is composed of fewer MT levels, reducing the reads (writes) to verify (update) a corresponding MT branch during a leaf node read (write). We illustrate this observation using Fig. 4. Figure 4(a) represents a classical single-root MT spanning 8 leaf nodes ($L_0 - L_7$), with the authentication path for leaf L_1 highlighted. In Fig. 4(b), the leaf nodes are allotted to 2 equal groups (G_0 and G_1), with an independent MT spanning each group and their roots (R_0 and R_1) maintained on the secure processor. As evident from the highlighted path, a smaller MT results in 1 less MT level read/write for authentication of L_1 . Generally, a k -ary MT with leaf nodes arranged in n groups achieves $\log_k n$ MT level reduction.

SMMT design: SMMT partitions the memory into memory block groups (MBGs), assigning an MT to each MBG, and maintaining the corresponding MT roots in an MT-root RAM on the secure processor. The individual MTs spanning each MBG are smaller than a single-root MT spanning the entire memory, substantially reducing the MT reads/updates, thereby decreasing (enhancing) NVM energy (system IPC). For n MBGs, the $\log_2 n$ most significant bits (MSBs) of the physical address provide the group index (G_i), utilized to select the appropriate MT root from the MT-root RAM.

Overhead: Whereas the advantages of SMMTs over classical single-root MTs scale logarithmically with the number of MBGs, it comes at the expense of linearly scaling on-chip MT-root RAM.

3.2.2 Dynamic multi-root Merkle Trees

Decentralized dynamic MMTs (DMMTs), as an alternative to SMMTs, provide the NVM energy and system IPC improvements of SMMTs without the significant overhead of processor MT-root RAM. DMMTs maintain a small MT over one frequently accessed MBG, and a larger MT spanning all other MBGs. As discussed below, ASSURE uses a low overhead memory access tracking architecture to translate the small MT across MBGs.

Observation: Practical workloads exhibit spatial and temporal locality for memory accesses, i.e., memory accesses are concentrated over a particular MBG (hot MBG henceforth). Therefore, maintaining a smaller hot MT over the hot MBG achieves SMMT-level reduction in the MT read/writes for authentication of a majority of the memory accesses. Since the remaining MBGs (cold MBGs) experience fewer memory accesses, maintaining a larger MT (cold MT henceforth) spanning the cold MBGs requires only one root at the expense of higher MT level traversals for a small fraction of the memory accesses. The DMMT thus stores only two secure roots (hot and cold roots), independent of the number of MBGs.

DMMT design: DMMTs maintain two MTs collectively spanning the memory, a hot MT spanning the hot MBG receiving majority of memory accesses and a cold MT covering the remaining MBGs, with both roots stored on the secure processor. Whereas a memory access to the hot MBG is authenticated with the smaller hot MT terminating at the hot root, an access to any cold MBG is authenticated with the larger cold MT concluding at the cold root.

Figure 5: DMMT organization with the hot MBG spanned by the hot MT, and a cold MT covering all remaining MBGs collectively, with their corresponding roots on the secure processor (R_{HOT} and R_{COLD}). Leaf L_1 (black) in the hot MBG requires lower MT level traversals than L_9 (yellow) in the cold MBG.

Figure 5 illustrates DMMT organization. The memory space, $T_{MEM}=16$, is divided into 4 MBGs, with MBG G_0 designated as the hot MBG and G_1, G_2 , and G_3 as the cold MBGs. Figure 5 also highlights the traversed nodes for authenticating (updating) L_1 (black) in the hot MBG, and L_9 (yellow) in the cold MBG. On an L_1 read, the recursive hash and compare procedure of authentication terminates at M_{20} , the hot MT root, which is maintained on the secure processor (R_{HOT}); however, on an L_9 read, the larger cold MT has greater MT traversal levels, concluding at the cold MT root (R_{COLD}). Similarly, on an L_1 write, only M_{10} in the memory and R_{HOT} are updated. Note that M_{20} and hence M_{30} are not updated during a write to the hot MBG.

Hot MBG prediction and update: DMMT requires effective hot MBG prediction to capture the majority of memory accesses. DMMT leverages the spatial and temporal locality of memory accesses in practical workloads for a simple, effective hot MBG prediction architecture. DMMT tracks the memory access count of each MBG over a period of P_{PRED} accesses, and designates the MBG that accounts for the maximum accesses as the hot MBG for the next P_{PRED} accesses. The access count for each MBG is reset after every P_{PRED} accesses for the next prediction cycle. For example, in Fig. 5, G_0 is initially considered the hot MBG, with 0 memory accesses to all the MBGs. Considering $P_{PRED}=16$, if MBG G_3 receives 10 accesses and G_0, G_1 , and G_2 each receives 2 accesses, then DMMT designates G_3 to be the hot MBG for the next P_{PRED} accesses.

Figure 6 illustrates the hot MBG prediction architecture of DMMT. The *AccessCount* RAM records the access count for each of n MBGs. During a leaf node read/write, the group index (G_i) of its corresponding MBG is obtained (refer Sec. 3.2.1), and the RAM content for that G_i is incremented by 1. The resultant sum (*ACOUNT*) is compared with the value of *MaxCount* register, which records the maximum access count within P_{PRED} accesses. If *ACOUNT* is greater than *MaxCount*, *MaxCount* is updated with *ACOUNT*, and the corresponding G_i is stored in the *NextHot* register that records which MBG has received the maximum accesses within a period. The *AccessCounter* is incremented on every memory access, and reset after P_{PRED} accesses, initiating a new cycle of prediction. When the counter output is 0, the *CurrentHot* register, which records the G_i of the current hot MBG, is updated with *NextHot* if they are unequal, and the new hot root is fetched from the memory.

Figure 6: Hot MBG prediction architecture for DMMT. We use $n=1024$ and $P_{PRED}=1024$ for our evaluations.

When the predicted hot MBG changes, DMMT updates the old hot root and its corresponding branch in the main memory. Subsequently, the sub-tree root spanning the new hot MBG is fetched, authenticated, and stored as the new hot root (R_{HOT}). In the example, when the hot MBG changes from G_0 to G_3 , the DMMT updates M_{20} with the latest value of R_{HOT} , followed by update of nodes M_{30} and R_{COLD} ; M_{23} , the root of the sub-tree covering G_3 , is fetched, authenticated, and assigned to R_{HOT} .

3.3 ASSURE authentication architecture

ASSURE synergistically integrates MMTs with SMACs for extending the NVM energy/lifetime/IPC improvements of SMACs to MMTs. ASSURE partitions each SMAC MMT node into k words for k -ary MMTs, and updates only that SMAC word corresponding to the modified child SMAC node. Although data SMACs utilize DEUCE modbits, ASSURE assigns a modbit to each SMAC node of the counter MMTs to identify its modified/unmodified child nodes.

Security: In ASSURE, SMACs do not alter the cryptographic hash algorithm of HMACs, maintaining full HMAC entropy; MMTs preserve the hash-and-compare flow with secure root architecture of single-root MTs. Hence, ASSURE preserves the security of the underlying MT memory authentication.

We evaluate the logic, memory, and latency overhead of ASSURE for a typical DEUCE-based encryption architecture with 32-bit line counters. We consider a 16GB data memory with 1GB counter memory, a 4-ary MMT over the counter memory, and HMAC based on SHA-1 that uses 128-bit codewords [7, 8].

Logic: We designed and synthesized the prediction architecture (refer Sec. 3.2.2) of DMMTs, which dynamically designates the hot MBG, for an estimated overhead of $\approx 2k$ 2-input nand gates.

Memory: In ASSURE, SMMT requires a $n \times 128$ -bit MT-root RAM, whereas DMMT requires a $n \times \log_2 P_{PRED}$ AccessCount RAM and a 2-root (hot/cold) RAM, where n is the total number of MBGs and P_{PRED} is the prediction cycle period. DMMT achieves optimum SMMT-level performance with $P_{PRED}=1024$, for $\approx n \times 10$ -bit RAM overhead, i.e., $12.8 \times$ less overhead than SMMT.

ASSURE also requires modbit storage in main memory for integration of the SMACs with MMTs. ASSURE assigns 1 modbit per 128-bit SMAC MMT node, resulting in $(1/128)$, i.e., $\approx 0.78\%$ overhead on the memory allocated to SMAC MMT nodes (ASSURE with SECRET [4] requires additional 1 modbit per 64-bit data word, i.e., $\approx 1.6\%$ memory overhead).

Latency: The main impact to authentication latency is the reset operation of the AccessCount RAM after P_{PRED} memory accesses. Since RAM does not provide RESET ports, it has to be explicitly cleared. For $n=1024$ and $P_{PRED}=1024$, DMMT requires 1.25kB RAM, which has an access latency of ≈ 1 ns, obtained using CACTI 5.3 [20] with low standby power transistors. Therefore, the AccessCount RAM reset incurs a latency of $n \times 1$ ns, i.e., 1024ns, every 1024 memory accesses; this translates to an amortized overhead of 1ns per memory access, which is insignificant compared to high access latencies of NVMs.

4. EVALUATION AND RESULTS

We evaluate ASSURE on a TLC RRAM architecture with integer and floating-point workloads from the SPEC CPU2006 [9] benchmarks. We consider a 4-ary MT for the evaluated authentication architectures: BMT [7] authentication (baseline), SMMT ASSURE, and DMMT ASSURE. Without exception, DEUCE is the underlying encryption framework; the use of SECRET, while beneficial to encryption, only marginally improves the results over DEUCE for memory authentication. We consider an MBG count n of 1024, and a prediction period P_{PRED} of 1024 memory access.

Simulation framework: We evaluate ASSURE for NVM energy, lifetime, and system IPC. For NVM energy evaluations, we perform trace-based simulations using NVMain [10]. We configure NVMain to reflect a 16GB single channel main memory with 2 ranks and eight x8 devices/rank. The memory controller performs first-ready-first-come-first-serve scheduling, with open page policy. The cell-level energy/latency parameters are provided in [17]. For lifetime evaluation, we use an in-house simulator that operates at the page level with a page size of 4kB. Along [17], we assume perfect wear leveling and a mean cell lifetime of 10^8 writes.

For system IPC evaluations, we use MARSS [12]. MARSS is configured to simulate a standard 4-core out-of-order system running at 3GHz. Each core has a private L1 I/D cache of 32kB (latency=2ns) and a private L2 cache of 128kB (latency=5ns). L3 is a shared write-back cache of 8MB (latency=20ns). The 16GB single-channel TLC RRAM main memory has 8 banks; the macro latency parameters are provided in [2]. We integrate a 128kB 32-way set-associative counter/MT metadata cache (32kB/core) [8] inside the memory controller for all the evaluated techniques. The HMAC computation is based on SHA-1 with 80-cycle latency [7, 21].

Workloads: To evaluate system performance, we use composite memory-intensive SPEC CPU2006 workloads, with each workload containing 4 benchmarks. Table 1 lists each workload with its constituent benchmarks and memory accesses (measured in L3 misses per kilo-instructions (MPKI)). We simulate a representative slice of 1 billion instructions for each workload.

Workload	Benchmarks	MPKI
WD ₁	h264ref, astar, milc, lbm	22.60
WD ₂	bzip2, gcc, sphinx3, xalancbmk	21.19
WD ₃	perlbench, soplex, bwaves, lbm	20.78
WD ₄	bzip2, gcc, mcf, omnetpp	19.09
WD ₅	perlbench, leslie3d, GemsFDTD, lbm	16.92
WD ₆	h264ref, sjeng, bwaves, povray	16.24
WD ₇	bzip2, mcf, namd, omnetpp	15.50
WD ₈	perlbench, soplex, milc, povray	13.44

Table 1: Workloads comprising of benchmarks from SPEC CPU2006 benchmark suite.

4.1 NVM energy

Figure 7 illustrates the impact of ASSURE on NVM energy for authentication, with SMMT ASSURE (DMMT ASSURE) reducing NVM energy, on average, by 59% (55%) over BMT authentication. ASSURE leverages the dual advantages of SMACs and MMTs. Whereas SMACs significantly reduce cell writes for data HMACs and each counter MMT node, MMTs decrease the number of MT node reads/writes. SMMTs achieve higher energy reduction than DMMTs, because all memory accesses encounter smaller MTs in SMMTs, whereas for DMMTs, memory accesses to the cold MBGs traverse a larger MT with more levels. DMMTs achieve $\approx 93\%$ of NVM energy reduction capabilities of SMMTs, with $\approx 12.8 \times$ smaller RAM overhead (AccessCount RAM and a 2-root (hot/cold root) RAM) than the SMMT MT-root RAM.

Figure 7: NVM energy of BMT, SMMT ASSURE, and DMMT ASSURE (normalized to baseline BMT) evaluated on SPEC CPU2006 benchmarks [9] using NVMain [10]. For TLC RRAM, SMMT ASSURE (DMMT ASSURE) reduces NVM energy, on average, by 59% (55%) over baseline BMT authentication.

4.2 Memory lifetime

Figure 8 illustrates the memory lifetime improvement offered by SMMT ASSURE (DMMT ASSURE) over baseline BMT. SMMT ASSURE (DMMT ASSURE) extends the memory lifetime, on average, by $2.36\times$ ($2.11\times$) over baseline BMT, through significant cell write reduction. Cell write reduction results in fewer programmed cells, thereby reducing the wear rate of memory. DMMT ASSURE offers $\approx 89\%$ of the lifetime improvement achieved by SMMT ASSURE, with DMMT ASSURE performing marginally worse than SMMT ASSURE due to a small fraction of memory accesses reaching the DMMT cold MBGs.

Figure 8: Memory lifetime of BMT, SMMT ASSURE, and DMMT ASSURE (normalized to baseline BMT) evaluated on SPEC CPU2006 benchmarks [9], utilizing an in-house simulator, using average cell lifetime of 10^5 writes until failure [17]. For TLC RRAM, SMMT ASSURE (DMMT ASSURE) improves memory lifetime, on average, by $2.36\times$ ($2.11\times$) over baseline BMT authentication.

4.3 System performance (IPC)

Figure 9 illustrates the impact of ASSURE on system performance. SMMT ASSURE (DMMT ASSURE) improves the system IPC, on average, by 11% (10%) over baseline BMT. ASSURE implements MMTs that diminish the number of MT node reads/writes by maintaining a smaller MT over the MBGs (hot MBG for DMMTs), thereby reducing bank contention between critical data/counter reads (writes) and MT node reads (writes). NVM systems are power constrained and update only a fixed number of cells per write slot [3]. SMACs in ASSURE enable multiple power-constrained concurrent writes in one write slot by reducing the effective number of cell updates per write, thereby diminishing the effective latency of authentication. The effectiveness of ASSURE becomes more evident for the high MPKI workloads (e.g., WD₁ and WD₂) that require more frequent authentication due to higher memory accesses.

Figure 9: System IPC of BMT, SMMT ASSURE, and DMMT ASSURE (normalized to baseline BMT) evaluated using workloads comprised of SPEC CPU2006 benchmarks [9] over MARSS [12]. For TLC RRAM, SMMT ASSURE (DMMT ASSURE) improves IPC, on average, by 11% (10%) over baseline BMT authentication.

4.4 Sensitivity: n and P_{PRED}

Figure 10 illustrates the impact of n and P_{PRED} over the effectiveness of DMMT in terms of average NVM energy, memory lifetime, and IPC, normalized to optimum n and P_{PRED} values of 1024 and 1024, respectively. Higher n values result in MBGs smaller than the spatial locality footprint of the program, suffering higher cold MT reads/writes, which is undesirable. Lower values of n result in larger hot MTs, resulting in increased MT read/writes.

Higher P_{PRED} leads to slower tracking of the memory access pattern change, resulting in higher cold MT reads/writes. Also, lower P_{PRED} values marginally affect DMMT performance for workloads with poor spatial locality, because lower P_{PRED} s lead to frequent updates of the changing hot MT roots and their corresponding branches.

Figure 10: Sensitivity of NVM energy, lifetime, and system IPC for DMMT to n and P_{PRED} , normalized to $n=1024$ and $P_{\text{PRED}}=1024$.

5. CONCLUSIONS

Memory authentication is key to ensuring data integrity in NVMs. However, in practice, it comes at the expense of increased NVM energy, degraded lifetime, and poor system IPC. ASSURE is the first work to address low cost NVM authentication. ASSURE integrates smart MACs (SMACs) and multi-root MTs (MMTs) to realize tamper-evident NVMs with low energy and improved lifetime as well as IPC. SMACs eliminate redundant HMAC computations of unmodified words on write-backs, reducing cell writes/NVM energy and improving lifetime. MMTs maintain multiple smaller MTs that collectively span the counter memory, reducing MT reads/writes for authentication, thereby reducing NVM energy, increasing lifetime, and improving system IPC. ASSURE outperforms state-of-the-art NVM authentication with 55% lower NVM energy, $2.11\times$ improved lifetime, and 10% better system IPC.

6. REFERENCES

- [1] B. C. Lee *et al.*, “Architecting phase change memory as a scalable DRAM alternative,” in *Proc. Intl. Symposium on Computer Architecture*, 2009.
- [2] C. Xu *et al.*, “Understanding the trade-offs in multi-level cell ReRAM memory design,” in *Proc. Design Automation Conference*, 2013.
- [3] V. Young *et al.*, “DEUCE: Write-efficient encryption for non-volatile memories,” in *Proc. Intl. Conference on Architectural Support for Programming Languages and Operating Systems*, 2015.
- [4] S. Swami *et al.*, “SECRET: Smartly EnCRypted energy Efficient non-volatile memories,” in *Proc. Design Automation Conference*, 2016.
- [5] A. Awad *et al.*, “Silent Shredder: Zero-cost shredding for secure non-volatile main memory controllers,” in *Proc. Intl. Conference on Architectural Support for Programming Languages and Operating Systems*, 2016.
- [6] G. E. Suh *et al.*, “Efficient memory integrity verification and encryption for secure processors,” in *Proc. Intl. Symposium on Microarchitecture*, 2003.
- [7] B. Rogers *et al.*, “Using address independent seed encryption and Bonsai Merkle Trees to make secure processors OS- and performance-friendly,” in *Proc. Intl. Symposium on Microarchitecture*, 2007.
- [8] A. D. Hilton *et al.*, “PoisonIvy: Safe speculation for secure memory,” in *Proc. Intl. Symposium on Microarchitecture*, 2016.
- [9] J. L. Henning, “SPEC CPU2006 benchmark descriptions,” in *ACM SIGARCH Computer Architecture News*, 2006.
- [10] M. Poremba and Y. Xie, “NVMain: An architectural-level main memory simulator for emerging non-volatile memories,” in *Proc. Computer Society Annual Symposium on VLSI*, 2012.
- [11] C.-K. Luk *et al.*, “Pin: Building customized program analysis tools with dynamic instrumentation,” in *Proc. Conference on Prog. Language Design and Implementation*, 2005.
- [12] A. Patel *et al.*, “MARSS: a full system simulator for multicore x86 CPUs,” in *Proc. Design Automation Conference*, 2011.
- [13] R. B. Lee, “Security basics for computer architects,” *Synthesis Lectures on Computer Architecture*, vol. 8, no. 4, 2013.
- [14] A. J. Menezes *et al.*, *Handbook of applied cryptography*. CRC press, 1996.
- [15] L. Jiang *et al.*, “A low power and reliable charge pump design for phase change memories,” in *Proc. Intl. Symposium on Computer Architecture*, 2014.
- [16] P. M. Palangappa and K. Mohanram, “CompEx: Compression-expansion coding for energy, latency, and lifetime improvements in MLC/TLC NVM,” in *Proc. Intl. Symposium on High Performance Computer Architecture*, 2016.
- [17] D. Niu *et al.*, “Low power multi-level cell resistive memory design with incomplete data mapping,” in *Proc. Intl. Conference on Computer Design*, 2013.
- [18] B. D. Yang *et al.*, “A low power phase change random access memory using a data-comparison write scheme,” in *Proc. Intl. Symposium on Circuits and Systems*, 2007.
- [19] S. Cho and H. Lee, “Flip-N-Write: A simple deterministic technique to improve PRAM write performance, energy and endurance,” in *Proc. Intl. Symposium on Microarchitecture*, 2009.
- [20] <http://www.hpl.hp.com/research/cacti/>.
- [21] T. Kgil *et al.*, “ChipLock: Support for secure microarchitectures,” in *Proc. Workshop on Architectural Support for Security and Anti-Virus*, 2005.