# ARSENAL: Architecture for Secure Non-Volatile Memories

Shivam Swami Kartik Mohanram
Department of Electrical and Computer Engineering, University of Pittsburgh, PA

## ABSTRACT

Whereas data persistence in non-volatile memories (NVMs) enables instant data recovery (IDR) in the face of power/system failures, it also exposes NVMs to data confidentiality and integrity attacks. Counter mode encryption and Merkle Tree authentication are established measures to thwart data confidentiality and integrity attacks, respectively, in NVMs. However, these security mechanisms require high overhead atomic security meta-data updates on every write-back in order to support IDR in NVMs. This increases memory traffic and negatively impacts system performance and NVM lifetime. **Architecture for Secure Non-Volatile Memories (ARSENAL)** is an IDR-preserving, low cost, high performance security solution that protects NVM systems against data confidentiality and integrity attacks. ARSENAL synergistically integrates (i) **Smart Writes for Faster Transactions (SWIFT)**, a novel technique to reduce the performance overhead of atomic security meta-data updates on every write-back, with (ii) **Terminal BMT Updates (TBU)**, a novel BMT-consistency-preserving technique, to facilitate IDR in the face of power/system failures. Our evaluations show that on average, ARSENAL improves system performance by 2.26×, reduces memory traffic overhead by 1.47×, and improves NVM lifetime by 2× in comparison to conventional IDR-preserving encryption+authentication.

## 1. Introduction and Motivation

Non-volatile memory (NVM) technologies like phase change memory (PCM), resistive RAM (RRAM), and spin-transfer torque RAM (STT-RAM) [1–4] have emerged as low power, high density, scalable alternatives for DRAM in main memories. Whereas data persistence in NVMs enables instant data recovery (IDR) in the face of power/system failures, data persistence also exposes NVMs to data confidentiality and/or integrity attacks [5–8].

Consistent with prior work on memory security [5–8], our **threat model** encompasses threats to data confidentiality and integrity in both the operational and powered-down states of the system; threats to system availability are excluded from the threat model. Furthermore, our trusted computing base (TCB) consists of the processor and core parts of the operating system; the external memory and peripherals are assumed to be untrusted. Along these lines, the stolen DIMM and bus snooping attacks are regarded as the two most common data confidentiality attacks on NVMs [5, 9]; data/bus tampering attacks are the two most common data integrity attacks [6, 7].

Counter mode encryption (CME) and integrity tree authentication [6–8, 10] are commonly used techniques to thwart data confidentiality and integrity attacks, respectively, in memory systems; this work uses address-independent seed encryption (AISE) [10] and Bonsai Merkle Tree (BMT) authentication [10] for data confidentiality and integrity, respectively, in the baseline system. Whereas AISE uses CME to ensure data confidentiality, BMT uses 1-level hash-based authentication to thwart spoofing/splicing integrity attacks and constructs a recursively hashed Merkle Tree over encryption counters to detect replay integrity attacks. AISE+BMT reserves a 7-bit minor counter and a 64-bit data message authentication code (DMAC) per 512-bit cache line; a 64-bit major counter is reserved per 4kB page. In practice, AISE+BMT is integrated on the processor-side memory controller (on the TCB), and frequently accessed security meta-data (counter/DMAC) is stored in an on-chip write-back (WB) counter cache to improve performance. On a read, a counter cache hit eliminates the extra memory accesses required to fetch the security meta-data. Similarly, on a write, the security meta-data is updated in the counter cache, eliminating the overhead of updating the security meta-data in the main memory.

Since a counter cache delays security meta-data updates in the main memory, it renders the security meta-data and program data partially inconsistent in the main memory. As a result, the program data residing in the main memory cannot be reliably decrypted and/or authenticated after a power/system failure, i.e., conventional AISE+BMT lacks the ability of NVMs to support IDR on power/system failures. Whereas atomic security meta-data updates (via a write-through counter cache and NVM logging [8, 11]) preserves IDR, this approach significantly increases memory writes, which deteriorates system performance and memory lifetime by 3.6× and 3×, respectively, in comparison to conventional AISE+BMT [12]. To the best of our knowledge, there is no security solution for NVMs that affords the benefits of AISE+BMT while simultaneously supporting IDR.

## 2. Contributions

We propose **Architecture for Secure Non-Volatile Memories (ARSENAL)** [12], an IDR-preserving, low cost, high performance security solution that augments AISE+BMT to protect NVM-based systems against data confidentiality and integrity attacks. ARSENAL makes two core contributions. First, ARSENAL integrates **Smart Writes for Faster Transactions (SWIFT)**, a novel technique that performs opportunistic, cache-line-level, pattern-based data compression to compress cache lines (i.e., plaintext) before encryption. SWIFT leverages the freed space to store the security meta-data required for encryption and authentication of the cache line. By opportunistically co-locating the ciphertext and its meta-data in a single 512-bit memory block, SWIFT not only ensures atomic updates to the ciphertext and its meta-data in a single write cycle, but also eliminates the extra read cycles required to fetch the meta-data on a read, thereby improving memory bandwidth and performance. For incompressible data, ARSENAL uses write-ahead logging (WAL) [11] to guarantee atomic updates to the security meta-data. Whereas SWIFT only employs WAL for incompressible data, AISE+BMT with IDR requires WAL on every write, imposing impractical performance and bandwidth overheads.

Second, ARSENAL performs **Terminal BMT Updates (TBU)** for data integrity protection across power/system failures. TBU leverages the key observation that the BMT can be reconstructed after a power/system failure as long as the BMT leaves and the on-chip BMT root are consistently updated with the ciphertext. Since BMT leaves are consistently updated via SWIFT/WAL, TBU has to only update the on-chip BMT root on every write-back. TBU thus replaces the expensive off-chip atomic BMT branch updates on every write-back by a single atomic on-chip BMT root update on every write-back.

This research was published in IEEE Computer Architecture Letters, 2018

| Security architecture | Instant data recovery | Memory traffic | System performance (IPC) | NVM lifetime |
|---|---|---|---|---|
| AISE+BMT+NDR | **no** | 1.15× | 0.70× | 0.46× |
| AISE+BMT+IDR | **yes** | 1.56× | 0.38× | 0.25× |
| **ARSENAL** | **yes** | **1.06×** | **0.86×** | **0.57×** |

Table 1: **Results for the 64-bit MAC system, normalized to the baseline.**

The consistently updated BMT root can be used to detect replay attacks that are carried out during the powered-down state of the system. This is done by reconstructing the BMT after a crash and verifying the reconstructed BMT root against the BMT root stored on the TCB. It is important to note that ARSENAL requires reconstruction of the BMT only when the system is restarted after a crash (i.e., power/system failure) and not when the system is restarted after a normal shutdown. *By shifting the overhead of atomically updating the modified BMT branch on every write-back during normal system operation to one-time post-crash BMT reconstruction and verification, ARSENAL provides a cost-effective solution to achieve IDR with AISE+BMT.*

# 3. Evaluation and Results

We evaluate ARSENAL using MARSS [13] and DRAMSim2 [14] for cycle-level, full-system simulation of the processor and memory. We configured MARSS to simulate a standard 4-core out-of-order system running at 3.2GHz. We modified MARSS to integrate a 128kB shared counter cache (32kB per core). In our analysis, encryption, decryption, and authentication incur latency of 80 cycles each. We also configured DRAMSIM2 to model a 16GB single-channel PCM based on [2]. Furthermore, to evaluate system performance in real-world scenarios, we used 9 composite workloads (WD) with each workload containing 4 benchmarks from the SPEC CPU2006 [15] benchmark suite.

**Evaluated techniques:** We consider a NVM system without any security support (encryption and/or authentication) as our **baseline**. We also compare ARSENAL with (i) AISE+BMT with no data recovery, i.e., **AISE+BMT+NDR** and (ii) AISE+BMT with instant data recovery, i.e., **AISE+BMT+IDR**.

Table 1 compares AISE+BMT+NDR, AISE+BMT+IDR, and ARSENAL in terms of IDR, memory traffic, system performance (measured in instructions per cycle, i.e., IPC), and NVM lifetime. The memory traffic, IPC, and lifetime results are normalized to the baseline. Results show that ARSENAL achieves IDR with the lowest memory traffic and the highest IPC and NVM lifetime.

**Memory traffic:** Figure 1 compares memory traffic overhead of AISE+BMT+IDR, AISE+BMT+NDR, and ARSENAL. We observe that AISE+BMT+IDR results in 56% increase in memory traffic due to atomic updates to the DMAC, counters, and BMT nodes. By delaying counter and BMT updates, conventional AISE+BMT+NDR increases memory traffic by only 15% (mainly due to the DMACs, since the DMACs are not stored in the counter cache [10,16]); however, AISE+BMT+NDR does not enable post-crash IDR, thereby compromising a key NVM property. In contrast, ARSENAL increases memory traffic by only 6%. By employing SWIFT, ARSENAL eliminates the overhead of reading/updating the DMAC and minor counter if the cache line being read/written is compressed/compressible. Additionally, due to TBU, ARSENAL does not require atomic updates to the modified BMT branch.

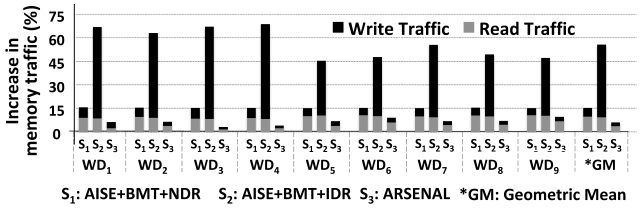


Figure 1: **Increase in total memory traffic for different security schemes.**

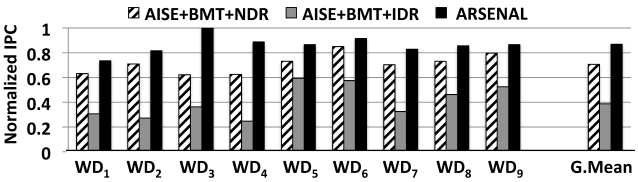


Figure 2: **IPC results show that AISE+BMT+NDR, AISE+BMT+IDR, and ARSENAL reduce IPC, on average, by 30%, 62%, and 14%, respectively.**

**System performance:** On average, in comparison to the baseline, ARSENAL reduces IPC by only 14%, whereas AISE+BMT+NDR (AISE+BMT+IDR) reduces IPC by 30% (62%). The IPC of ARSENAL is better than both AISE+BMT+IDR and AISE+BMT+NDR due to (i) SWIFT, which saves the mandatory additional read/write of the off-chip DMAC by opportunistically storing the DMAC alongside the ciphertext as a single 512-bit memory block and (ii) TBU, which eliminates atomic BMT updates that increase write latency and memory traffic of AISE+BMT+IDR. Further, we observe a strong correlation between memory traffic reduction and IPC improvement. For example, ARSENAL has the lowest memory traffic for $WD_3$; consequently the IPC for $WD_3$ is the highest.

# 4. Conclusions

ARSENAL is a holistic solution for data confidentiality, data integrity, and IDR in NVM systems. ARSENAL synergistically integrates SWIFT and TBU to ensure low-overhead atomic security meta-data updates for IDR. In comparison to conventional encryption+authentication approaches that support IDR, ARSENAL improves IPC by 2.26×, reduces memory traffic by 1.47×, increases NVM lifetime by 2×.

# 5. References


[1] *International Technology Roadmap for Semiconductors*, 2011.

[2] B. C. Lee *et al.*, "Architecting phase change memory as a scalable DRAM alternative," in *Proc. ISCA*, 2009.

[3] H. Akinaga and H. Shima, "Resistive random access memory (ReRAM) based on metal oxides," *Proc. of the IEEE*, 2010.

[4] E. Kültürsay *et al.*, "Evaluating STT-RAM as an energy-efficient main memory alternative," in *Proc. ISPASS*, 2013.

[5] V. Young *et al.*, "DEUCE: Write-efficient encryption for non-volatile memories," in *Proc. ASPLOS*, 2015.

[6] S. Gueron, "A memory encryption engine suitable for general purpose processors," *IACR Cryptology ePrint Archive*, 2016.

[7] M. Taasori *et al.*, "VAULT: Reducing paging overheads in SGX with efficient integrity verification structures," in *Proc. ASPLOS*, 2018.

[8] S. Liu *et al.*, "Crash consistency in encrypted non-volatile main memory systems," in *Proc. HPCA*, 2018.

[9] S. Swami *et al.*, "SECRET: Smartly encrypted energy efficient non-volatile memories," in *Proc. DAC*, 2016.

[10] B. Rogers *et al.*, "Using address independent seed encryption and Bonsai Merkle Trees to make secure processors OS- and performance-friendly," in *Proc. MICRO*, 2007.

[11] C. Mohan *et al.*, "ARIES: A transaction recovery method supporting fine-granularity locking and partial rollbacks using write-ahead logging," *ACM TODS*, 1992.

[12] S. Swami and K. Mohanram, "ARSENAL: Architecture for secure non-volatile memories," *IEEE Computer Architecture Letters*, 2018.

[13] A. Patel *et al.*, "Marss-x86: A Qemu-based micro-architectural and systems simulator for x86 multicore processors," in *Proc. DAC*, 2011.

[14] P. Rosenfeld *et al.*, "DRAMSim2: A cycle accurate memory system simulator," *IEEE CAL*, 2011.

[15] https://www.spec.org/cpu2006/.

[16] T. S. Lehman *et al.*, "PoisonIvy: Safe speculation for secure memory," in *Proc. MICRO*, 2016.