ASSET: Architectures for Smart Security of Non-Volatile Memories

Shivam Swami and Kartik Mohanram
Department of Electrical and Computer Engineering, University of Pittsburgh, PA
shs173@pitt.edu kartik.mohanram@gmail.com

ABSTRACT

Computing systems that integrate advanced non-volatile memories (NVMs) are vulnerable to several security attacks that threaten (i) data confidentiality, (ii) data availability, and (iii) data integrity. This paper proposes Architectures for Smart Security of NVMs (ASSET), which integrates five low overhead, high performance security solutions—SECRET [1], COVERT [2], ACME [3], ARSENAL [4], and STASH [5]—to thwart these attacks on NVM systems. SECRET is a low cost security solution that employs counter mode encryption (CME) for data confidentiality in multi-/triple-level cell (i.e., MLC/TLC) NVMs. COVERT and ACME complement SECRET to improve system availability of CME. ARSENAL integrates CME and Bonsai Merkle Tree (BMT) authentication to thwart data confidentiality and integrity attacks, respectively, in NVMs and simultaneously enables instant data recovery (IDR) on power/system failures. Finally, STASH is the first comprehensive end-to-end security architecture for state-of-the-art smart hybrid memories (SHMs). STASH integrates (i) CME for data confidentiality, (ii) page-level MT authentication for data integrity, (iii) recovery-compatible MT updates to withstand power or system failures, and (iv) page-migration friendly security meta-data management. This paper thus addresses the core security challenges of next-generation NVM systems.

Keywords: Non-volatile memories, smart hybrid memories, confidentiality, integrity, availability, encryption, authentication

1. Introduction

Non-volatile memory (NVM) technologies such as phase change memory (PCM) [6], resistive RAM (RRAM) [7], and 3D X-Point [8] have emerged as promising replacement candidates for DRAM due to their better scalability, higher data density, and lower leakage power. These resistance-class NVMs store data by modulating the resistance of the storage material. Due to a large separation between the lowest and the highest resistance states, these NVMs also support multi-/triple-level cell (MLC/TLC) operation, i.e., they offer the ability to store two/three logical bits per physical cell, increasing data density and driving down cost.

NVM Security: Whereas PCM/RRAM/3D X-Point offer several advantages over DRAM, computing systems that integrate these advanced NVMs are vulnerable to security attacks that threaten (i) data confidentiality, (ii) data availability, and (iii) data integrity. Data confidentiality attacks aim to obtain secret data stored in the system [9–14]. Data availability attacks seek to make the memory system unavailable for authorized users [15, 16]. Finally, data integrity attacks refer to any adversarial corruption or tampering of data [16–19]. In addition to these direct attacks on data confidentiality, availability, and integrity, memory systems are also vulnerable to side-channel attacks that exploit memory access patterns to obtain secret encryption/authentication keys [20–22]. This paper proposes ASSET, which integrates five low overhead, high performance security solutions—SECRET [1], COVERT [2], ACME [3], ARSENAL [4], and STASH [5]—to thwart these attacks on NVMs.

SECRET [1]: Data encryption preserves data confidentiality in NVMs; however, encryption renders cell flip reduction techniques like data comparison write (DCW, i.e., read-modify-write) [23] and flip-n-write (FNW) [24]—employed to reduce cell failures in write-limited NVMs—ineffective in practice [12]. This reduces NVM lifetime and increases write energy/latency. *Smartly EnCRypted Energy efficienT (SECRET)* NVMs synergistically integrate zero-based partial writes with XOR-based energy masking to realize low overhead data encryption for MLC/TLC NVMs. Our simulations on an MLC (TLC) resistive RAM (RRAM) architecture across SPEC CPU2006 workloads demonstrate that for comparable overhead, SECRET reduces write energy by 3× (1.6×), latency by 1.23× (1.5×), and improves memory lifetime by 1.2× (1.25×) over state-of-the-art DEUCE [12].

COVERT [2] and ACME [3]: State-of-the-art encryption techniques [9, 11, 12] are based on the principles of counter mode encryption (CME), which associates a counter with each cache line and uses this counter along with the memory address of the cache line and a secret key to encrypt the cache line on a memory write. However, CME suffers from the counter overflow problem, in which a counter overflow mandates full memory re-encryption with a new secret key, causing the system to freeze for the duration of full memory re-encryption. Counter overflow renders CME vulnerable to the denial of memory service (DoMS) attacks that threaten system availability. *Counter OVErflow ReducTion (COVERT)* is an encryption solution that addresses the counter overflow problem in CME by performing on-demand memory allocation to the fast-growing counters, while also retaining the area/performance benefits of small counters. *Advanced Counter Mode Encryption, i.e., ACME*, complements COVERT by leveraging the underlying wear leveling architecture—employed to improve NVM endurance [6, 25–27]—to perform counter write leveling (CWL). We evaluate ACME+COVERT for PCM using (i) a trace-driven memory simulator that integrates the Intel Pin [28] binary instrumentation tool and (ii) the MARSS [29] full-system simulator on SPEC CPU2006 benchmarks [30]. Results show that for the system availability of 99.999%, ACME+COVERT not only requires 50% lower counter memory overhead, but also improves system performance by 20% in comparison to classical CME with 64-bit counters.

ARSENAL [4]: Although NVM encryption ensures data confidentiality, it does not guarantee data integrity, which is the ability to detect adversarial tampering of (i) stored data and/or (ii) data transactions to/from memory [16, 17, 31]. Bonsai Merkle Tree (BMT) authentication is the established measure to thwart data integrity attacks in NVMs [31, 32]. However, BMT requires high overhead atomic security meta-data updates on every write-back in order to support instant data recovery (IDR) in NVMs. This increases memory traffic and negatively impacts system performance and memory lifetime. 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 (measured in IPC) by 4×, reduces memory traffic overhead by 1.88×, and improves memory lifetime by 3.5× in comparison to conventional IDR-preserving 128-bit encryption+authentication.

978-1-7281-3391-1/19/$31.00 ©2019 IEEE
DOI 10.1109/ISVLSI.2019.00070

STASH [5]: Smart hybrid memories (SHMs) that integrate NVM, DRAM, and processor logic can provide high bandwidth, low memory latency, and high memory density to meet the needs of future high-performance computing systems. However, the unsecure DRAM, NVM, and/or memory buses in SHMs are vulnerable to data confidentiality attacks (e.g., memory scanning and bus snooping) [10, 12, 16, 32], data integrity attacks (spoofing/splicing/replay attacks) [18,20,21,31], and side-channel attacks (e.g., access-pattern based attacks) [20, 21] that must be addressed prior to commercialization. *SecuriTy Architecture for Smart Hybrid memories (STASH)* is the first comprehensive end-to-end security solution that integrates (i) CME for data confidentiality, (ii) page-level MT authentication for data integrity, (iii) recovery-compatible MT updates to withstand power/system failures, and (iv) page-migration friendly security meta-data management. For security guarantees equivalent to state-of-the-art ObfusMem [21], STASH reduces memory overhead by 12.7×, improves system performance by 65%, and increases NVM lifetime by 5×.

The rest of this paper is organized as follows. Section 2 provides necessary background in NVM security and motivates ASSET. Section 3 describes various contributions (SECRET, COVERT, ACME, ARSENAL, and STASH) of ASSET, along with evaluation and results. Finally, section 4 presents conclusions and directions for future research.

2. Background and motivation

This section discusses the NVM threat model, the role of memory encryption and authentication in NVM security, and outlines the challenges of state-of-the-art NVM security solutions to motivate ASSET.

2.1 Threat model

The security of modern computing systems is based on the three cornerstone properties of confidentiality, integrity, and availability [16]. As in prior studies on memory security, we assume that the trusted computing base (TCB) consists of the processor and core parts of the operating system (e.g., security kernels); external memory and peripherals are assumed to be untrusted [9, 11–13, 31]. Our threat model encompasses attacks on data confidentiality and integrity in both the operational and powered-down system states; data availability attacks are covered only in the operational system state. This threat model is extended to include the on-module processing logic in smart hybrid memories (discussed in section 3.4).

2.1.1 Data confidentiality attacks

Past research [9–12] on NVM security has recognized the stolen DIMM and bus snooping attacks as the two most common security attacks to data confidentiality in NVMs. Unlike DRAM, where only specialized cold boot attacks [33] can potentially retrieve data after power down, data retrieval from powered-down NVMs is much easier due to data persistence. It is widely accepted that data confidentiality attacks can be thwarted by implementing data encryption in the TCB.

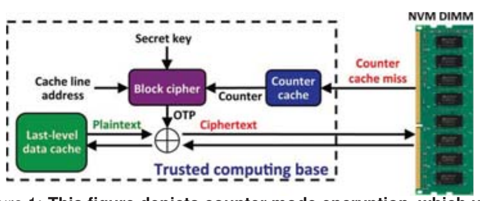

Figure 1: **This figure depicts counter mode encryption, which uses a counter, line address, and secret key to generate a one-time pad (OTP). During encryption (decryption), the OTP is XORed with the plaintext (ciphertext) to generate the ciphertext (plaintext).**

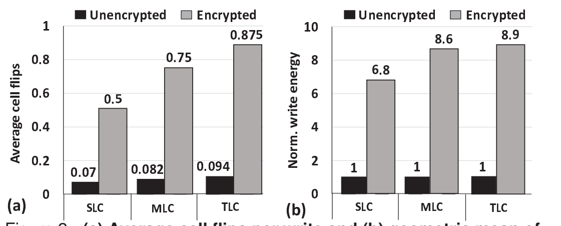

Figure 2: **(a) Average cell flips per write and (b) geometric mean of write energy (normalized to unencrypted data) for SPEC CPU2006 benchmarks for SLC/MLC/TLC RRAM with DCW.**

Recent research [12, 13] advocates the use of **counter mode encryption (CME)** as a secure memory encryption technique to thwart data confidentiality attacks on NVMs. In CME, a block cipher is used to encrypt a seed with a secret key to produce a one-time pad (OTP). This OTP is bitwise XORed with the plaintext to generate the ciphertext. During decryption, the same OTP is XORed with the ciphertext to obtain the plaintext. The spatial and temporal exclusivity of the OTP is critical for the security of CME and requires that (i) the OTPs should be unique for different cache lines and (ii) the OTPs for a particular cache line should be unique for every write. These unique OTPs are generated from unique seeds that have two components: (i) the cache line address (for spatial exclusivity) and (ii) a counter, which is incremented on each write (for temporal exclusivity). To reduce on-chip memory overhead in CME, the counters are stored in main memory and cached in an on-chip counter cache to improve performance [12, 32, 34]. Figure 1 shows CME in the presence of a counter cache.

Challenge: Figure 2 illustrates the impact of encryption on average cell flip rate and write energy for SLC/MLC/TLC RRAM. As shown in the figure, encryption increases cell flip rate (write energy) by $> 9\times$ ($8\times$) for MLC/TLC RRAM, rendering cell flip reduction techniques like DCW [23] ineffective in practice. Hence, the challenge is to develop a low write overhead encryption architecture for NVMs without compromising data confidentiality.

2.1.2 Data availability attacks

A counter overflow in CME is handled by changing the secret key to prevent reuse of OTPs [17]. However, since the same secret key is shared by every cache line, a change of secret key requires the entire memory to be re-encrypted, causing the system to freeze for the duration of full memory re-encryption.

Since CME requires full memory re-encryption upon counter overflow, NVM-based systems that employ CME become vulnerable to denial of memory service (DoMS) attacks that threaten system availability. In a DoMS attack, a malicious application can render the memory system unavailable to other applications by forcing frequent full memory re-encryption due to counter overflow. DoMS attacks can be easily engineered using cache eviction and ordering instructions (as shown below) that can be executed in non-administrator mode to constantly write to the same cache line in main memory, forcing its counter to overflow.

```
size_t x = 0;
while(1) {
    x = x + 1;
    asm volatile ("clflush (%0)" :: "r"(&x));
    asm volatile ("mfence");
}
```

Challenge: To reduce counter overflow and increase system availability, large counters are used for encryption, since they do not overflow frequently. However, large counters increase the memory overhead of CME and result in poor system performance due to frequent counter cache misses. CME thus imposes heavy overheads on memory, performance, and system availability in practice.

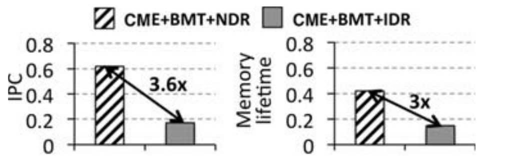

Figure 3: **Comparison of IPC and memory lifetime of CME+BMT+NDR and CME+BMT+IDR. The results are normalized to a system that does not perform encryption or authentication.**

2.1.3 Data integrity attacks

Data integrity attacks refer to the tampering of data in order to compromise the security of the computing system on which the data is stored [11, 16, 17, 19, 32]. Data integrity attacks are classified into spoofing, splicing, and replay attacks. Arbitrary data tampering by the adversary constitute spoofing attacks. These attacks can potentially disrupt the normal system operation or reveal confidential information stored in the system. Unauthorized copying (also swapping) of data from one memory address to another constitute splicing attacks. Such an attack may be viewed as a spatial permutation of memory blocks. Finally, Replacing a memory block's data with a valid older value constitute replay attacks. A memory block located at a given address is recorded and inserted at the same address at a later point in time. Such an attack may be viewed as a temporal permutation of a memory block, for a specific memory location.

Bonsai Merkle Tree (BMT) is a main memory authentication approach to thwart data integrity attacks [18,31,35]. BMT protects each cache line using a single-level data message authentication code (DMAC), which is a 64-bit or a 128-bit hashed key value of the ciphertext, counter, and cache line address. Although single-level DMACs are robust against spoofing and splicing attacks, they are ineffective against replay attacks wherein an adversary replaces the data and counter with their older values [17, 18]. To protect memory from replay attacks, BMT maintains a hierarchical tree structure of hashes, with the counters as its leaf nodes. The integrity of the fetched counter is verified by computing the corresponding chain of hashes up to the BMT root, which is maintained on the processor-side memory controller.

Challenge: To preserve the instant data recovery (IDR) property of NVMs, security metadata (i.e., BMT) must be atomically updated on every write-back. Due to BMT's hierarchical tree structure, ensuring atomic BMT updates incurs high memory traffic, which negatively impacts performance and reduces NVM lifetime. Figure 3 compares the system performance (measured in instructions per cycle, i.e., IPC) and memory lifetime of CME+BMT+IDR (i.e., CME+BMT with a write-through counter cache and write-ahead logging (WAL) [36] for IDR) with conventional CME+BMT+NDR (i.e., CME+BMT with no data recovery (NDR)). The results are normalized to an ideal system that does not employ encryption and/or authentication (refer section 3.3 for simulation details). Our results show that CME+BMT+IDR reduces IPC by 3.6× and reduces memory lifetime by 3× in comparison to CME+BMT+NDR.

2.1.4 Side-channel attacks

In addition to these direct attacks on data confidentiality, availability, and integrity, memory systems are also vulnerable to side-channel attacks that exploit memory access patterns to obtain secret encryption/authentication keys [20–22]. Concealing the true memory access patters by employing techniques like Oblivious RAM (ORAM) [37] thwarts access-pattern based side-channel attacks.

Challenge: ORAM obfuscates information about (i) the address accessed, (ii) the access type (read or write), and (iii) the data being read/written; however, it increases memory traffic by 10× and deteriorates performance by 4× [20, 21, 37].

3. Contributions

The goal of ASSET is to address the challenges (discussed in section 2) of NVM security, without compromising security, system performance, system availability, and NVM lifetime.

3.1 SECRET

SECRET [1] targets the high cell flip rate of encrypted MLC/TLC NVMs. To reduce cell flips, SECRET synergistically integrates **smart encryption** with **XOR-based energy masking**, without compromising the security of the underlying encryption technique.

Smart encryption: During the write-back operation of a cache line, the majority of the words remain unmodified [11, 12]. Smart encryption preserves these words in their previous encrypted states, without compromising the security of the data. In this work, we refer to this as word-level DCW. We perform word-level DCW to prevent undesired cell flips by blocking the re-encryption of the unmodified words. To further reduce the encryption penalty, we leverage the fact that a significant fraction of the plaintext written to the memory is zero [38], and prevent the re-encryption of this zero data. We use a one-bit zero-flag per word to track zero-words (i.e., words with only zeros) in a cache line and maintain zero-words in their last encrypted states, saving the write overhead of re-encrypting zero-words. Figure 4 illustrates smart encryption.

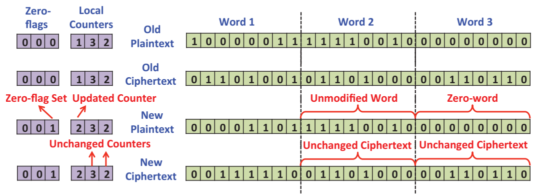

Figure 4: **Smart encryption prevents the re-encryption of unmodified words (word 2) and zero-words (word 3). The zero-flags track zero-words in a cache line; zero-flag = 1 for word 3 in the figure.**

Energy masking: Following smart encryption, we perform write optimization by filtering the encrypted words (i.e., the ciphertext) through energy masks. For an overhead of one bit (energy-flag) per word, XOR-based energy masks transform high energy states in the ciphertext into low energy states, reducing the overall write energy of the cache line. Both the zero-flag and the energy-flag are stored in encrypted state in the memory to ensure the security of the data.

Evaluation and results: SECRET is evaluated on MLC and TLC RRAM architectures using the NVMain [39] memory simulator on memory traces from the SPEC CPU2006 benchmark suite [30]. We consider advanced encryption standard (AES-based) CME as our baseline. SECRET is compared to state-of-the-art encryption techniques, namely block-level encryption (BLE) [11] and dual counter encryption (DEUCE) [12], which perform cache line encryption at a granularity of 128 bits and 16 bits, respectively. Our results are summarized in Table 1. As shown in the table, SECRET outperforms state-of-the-art NVM encryption solutions, with the lowest write energy and latency, as well as the highest lifetime.

NVM	Encryption technique	Energy reduction	Latency reduction	Lifetime improvement	Memory overhead
MLC RRAM	BLE	40%	23%	35%	1.56%
	DEUCE	40%	17%	36%	6.25%
	SECRET	80%	37%	63%	6.25%
TLC RRAM	BLE	33%	31%	18%	1.56%
	DEUCE	40%	23%	24%	6.25%
	SECRET	63%	49%	56%	7.84%

Table 1: **Energy, latency, lifetime improvements, and memory overhead of BLE, DEUCE, and SECRET for MLC/TLC NVMs.**

3.2 COVERT and ACME

COVERT [2] and ACME [3] are practical solutions to address the low system availability of CME. At its core, COVERT employs dynamic counters (DYNAMO henceforth) to reduce frequent full memory re-encryption due to small-sized counters. DYNAMO leverages the fact that a significant fraction of memory provisioned for error correction remains unutilized till very late in memory lifetime [26, 40, 41]. DYNAMO repurposes unused error correction memory cells to the overflowing counters, thereby delaying the mandatory full memory re-encryption on a counter overflow and improving system availability. It is important to note that COVERT is a drop-in replacement for classical CME, and does not compromise the security of the underlying CME (i.e., COVERT preserves CME requirements of (i) encryption inside a secure processor and (ii) spatial/temporal exclusivity of the OTP).

Since the availability of idle error correction memory cells reduces as the memory ages, the ability of COVERT to delay counter overflow decreases with time. As a result, COVERT becomes less effective against DoMS attacks later in the memory lifetime. ACME complements COVERT to thwart DoMS attacks without relying on limited error correction memory cells. At its core, ACME leverages the underlying wear leveling architecture—employed to improve NVM endurance [25, 26]—to perform counter write leveling (CWL). CWL associates counters with physical addresses (PAs) instead of logical addresses (LAs) such that when a frequently written cache line is translated from one PA to another PA for wear leveling, its associated counter is also remapped, leading to a distribution of writes across counters. Multiple counters together track the number of writes seen by a frequently written cache line, thereby delaying counter overflow; delaying counter overflow improves system availability due to delayed full memory re-encryption.

Figure 5 contrasts CME and ACME. When cache line ‘D’ is translated from PA 40 to PA 50 using CME (figure 5(a)), its associated counter, i.e., CTR_{D} remains unchanged and continues to grow with writes to cache line ‘D’, resulting in early overflow. In contrast, when cache line ‘D’ is translated from PA 40 to PA 50 using ACME (figure 5(b)), its associated counter is changed from CTR_{40} to CTR_{50}. This results in sharing of writes to cache line ‘D’ between CTR_{40} and CTR_{50}. CWL thus increases the time to counter overflow, thereby delaying the time to full memory re-encryption.

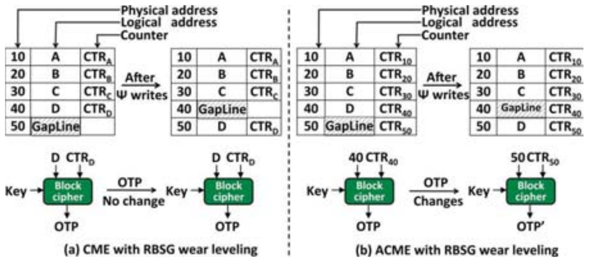

Figure 5: **This figure contrasts OTP generation of CME and ACME, in the presence of Region-Based Start-Gap (RBSG) wear leveling. (a) OTP generation for CME remains unaffected after cache line ‘D’ is remapped. (b) OTP generation for ACME requires the new PA and its corresponding counter value after cache line ‘D’ is remapped. In this work, the GapLine moves after every ψ writes (ψ = 64).**

Evaluation and results: We evaluate ACME on a 16GB PCM architecture using (i) a trace-driven memory simulator that integrates the Intel Pin [28] binary instrumentation tool and (ii) the MARSS [29] full-system simulator on SPEC CPU2006 benchmark suite [30]. The system performance is evaluated in instructions per cycles (IPC) and the system availability is derived from the ratio of the total up time to the aggregate of total up and down time. Following [2], we assume total down time of 1 minute for full memory re-encryption. We compare ACME+COVERT with three state-of-the-art encryption architectures namely split-CME [17, 32], MECU [9], and DEUCE [12]. Table 2 summarizes the counter memory overhead, system performance (in IPC) and system availability of the evaluated techniques. Results show that ACME+COVERT provides the optimal trade-off between memory overhead, system performance, and system availability in practice.

Technique	Counter size	IPC	System availability	Memory overhead
Split-CME	64 bits	0.65	99.999%	1.56%
MECU	16 bits	0.84	85.714%	3.12%
DEUCE	28 bits	0.79	99.995%	5.46%
ACME+COVERT	24 bits	0.82	99.999%	4.88%

Table 2: **Energy, latency, lifetime improvements, and memory overhead of BLE, DEUCE, and SECRET for MLC/TLC NVMs.**

Further, to validate the robustness of ACME+COVERT to availability attacks, we launched a DoMS attack on a Linux machine with an Intel Core i3 CPU and 16GB main memory. The attack invokes `clflush` and `mfence` instructions to sidestep all levels of cache and constantly writes to the same LA until the counter associated with that LA overflows. We observed that when subject to the DoMS attack, the ACME+COVERT-based system provides 99.9% availability in contrast to a classical CME-based system that is rendered non-operational.

3.3 ARSENAL

ARSENAL [4] addresses the high performance penalty and reduced NVM lifetime caused due to atomic security metadata updates required to preserve IDR in NVMs. ARSENAL employs **Smart Writes for Faster Transactions (SWIFT)** and **Terminal BMT Updates (TBU)** to augment CME+BMT [18, 32] to protect NVMs against data confidentiality and integrity attacks.

SWIFT: Swift is 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 storing the security meta-data alongside the ciphertext (i.e., encrypted cache line) as a single memory block, SWIFT enables atomic updates to the security meta-data in a single memory access, resulting in improved performance and reduced memory traffic. For incompressible data, ARSENAL uses write-ahead logging (WAL) [36] to guarantee atomic updates to the security meta-data.

TBU: ARSENAL integrates 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 leaf nodes and the on-chip BMT root are consistently updated with the ciphertext. Since BMT leaf nodes are consistently updated via SWIFT/WAL, TBU has to only update the on-chip BMT root on every write-back.

Evaluation and results: We evaluate ARSENAL using MARSS [29] and DRAMSim2 [42] for cycle-level, full-system simulation of the processor and memory, respectively. We modified MARSS to integrate a 128kB shared counter cache (32kB per core). Encryption, decryption, and authentication incur latency of 80 cycles each [35]; the default MAC size is 128 bits. We also configured DRAMSIM2 to model a 16GB single-channel PCM based on [6]. 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 [30] benchmark suite.

Security architecture	Instant data recovery	Memory traffic	System performance (IPC)	NVM lifetime
CME+BMT+NDR	**no**	1.28×	0.62×	0.42×
CME+BMT+IDR	**yes**	2.13×	0.17×	0.14×
ARSENAL	**yes**	**1.13**×	**0.73**×	**0.49**×

Table 3: **Results for the 128-bit MAC system, normalized to the baseline.**

We consider a system without encryption/authentication as our **baseline**. We also compare ARSENAL with (i) CME+BMT with no data recovery (CME+BMT+NDR) and (ii) CME+BMT with data recovery (CME+BMT+IDR). Table 3 compares each evaluated technique in terms of IDR, memory traffic, system performance (measured in 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.

3.4 STASH

Whereas emerging NVMs are low power, dense, scalable alternatives to DRAM, the high latency and low endurance of these NVMs limit the feasibility of NVM-only memory systems [26,43–45]. Smart hybrid memories (SHMs) that integrate smart NVM (e.g., Intel Optane [46]), smart DRAM (e.g., Micron HMC [47]), and on-module processor logic are an efficient means to bridge the latency–endurance gaps between NVM-only and DRAM-only memory systems. However, these SHMs are vulnerable to data confidentiality, integrity, and side-channel attacks that can be executed on the unsecure NVM, DRAM, and/or memory buses. STASH [5] is the first comprehensive end-to-end security architecture for SHMs that integrates (i) CME for data confidentiality, (ii) low overhead page-level MT (**PMT**) for data integrity, (iii) recovery-compatible MT updates (**RECOUP**) to withstand power/system failures, and (iv) page-migration-friendly security meta-data management (**PACT**).

Strawman Security Architecture (SSA): To thwart data confidentiality, data integrity, and side-channel attacks in SHMs, a strawman security architecture for SHMs (SHM-SSA) can be derived by extending state-of-the-art security solutions, e.g., ObfusMem [21] and InvisiMem [20]. Figure 6 shows a comprehensive end-to-end security framework using SHM-SSA. To ensure instant data recovery (IDR) upon system restart or in the face of power/system failures, SHM-SSA ensures data confidentiality and integrity in both the operational and powered-down system states. SHM-SSA employs (i) 128-bit AES-based CME for data confidentiality (Fig. 6(a)), (ii) 128-bit BMT authentication for data integrity (Fig. 6(b)), (iii) access pattern obfuscation to thwart snooping attacks on address and command buses (Fig. 6(c)), and (iv) Galois/Counter Mode (GCM) authenticated encryption to thwart tampering attacks on memory buses (Fig. 6(d)). Memory bus encryption and authentication is enabled due to the crypto engines present on the secure logic layers of smart DRAM and smart NVM.

Whereas SHM-SSA ensures end-to-end security for SHMs, SHM-SSA imposes heavy overheads on memory, performance, and NVM lifetime. Our system-level simulations of SHM-SSA (for a system that integrates a 2GB HMC [47] (as DRAM cache) with a 32GB smart TLC PCM [48]) show that in comparison to unsecure SHM, SHM-SSA reduces system performance (measured in instructions per cycles (IPC)) by $> 2\times$ and NVM lifetime by $> 6\times$. The key bottlenecks for the low IPC and NVM lifetime of SHM-SSA are (i) latency-intensive BMT authentication, (ii) poor DRAM utilization due to high overhead of the security meta-data, and (iii) high cell flip rate of the hashes/encrypted data that is updated on every write to support IDR. STASH is a refinement of SHM-SSA that integrates PMT, RECOUP, and PACT to overcome security, IDR, and page migration overheads, respectively, of SHM-SSA.

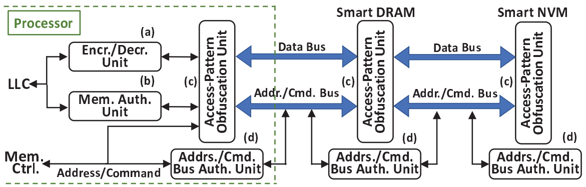

Figure 6: **This figure illustrates the end-to-end security architecture of SHM-SSA. Only the secure processing logic of smart DRAM and smart NVM are shown in the figure.**

PMT: PMT replaces classical cache-line-level authentication to secure the SHM from data integrity attacks. PMT leverages 4kB page granularity data migration between DRAM and NVM to reduce Merkle Tree (MT) authentication overheads for thwarting data tampering attacks. BMT [18] framework is the norm for memory authentication that is adopted by all state-of-the-art memory security techniques [21,31,32]. BMT requires a 128/64-bit DMAC per 512-bit cache line along with an MT constructed over encryption counters. In contrast, PMT maintains a 128/64-bit MAC per 4kB page and constructs an MT by recursively hashing these page-level MACs (PMACs), providing equivalent security guarantees for significantly lower memory and performance overheads.

RECOUP: RECOUP is an IDR-preserving solution that performs selective meta-data (counters, BMT root, etc.) updates to tolerate power/system failures in SHMs. To improve performance, it is common practice to employ an on-chip write-back counter cache to delay security meta-data updates in memory [18,32]. This renders the meta-data residing in the SHM partially stale and unsynchronized with the ciphertext. As a result, the SHM cannot reliably decrypt and/or authenticate the ciphertext in the event of a power/system failure. SHM-SSA employs a write-through (WT) counter cache to support IDR in SHMs. This significantly increases NVM writes, since every cache line write also requires multiple consistent MT updates. RECOUP leverages a key observation that IDR can be supported by consistently updating only (a) the MT root in the TCB of the smart DRAM and (b) the modified MT leaf (i.e., data/counter) in the smart NVM, thereby eliminating unnecessary MT re-computations and reducing high overhead NVM writes.

PACT: PACT supports low overhead page migration is SHMs. Since state-of-the-art security solutions [17,18,20,21,32] are designed to operate at only cache-line granularity, the meta-data overhead (>1kB per 4kB page) of these solutions on page migration from smart NVM to smart DRAM increases memory traffic and reduces effective DRAM capacity. PACT addresses this problem by (i) transferring only the required PMT meta-data (16 bytes per 4kB page) to DRAM and (ii) caching the PMT meta-data of the migrated page in a PMAC cache on the logic layer of the smart DRAM. By eliminating bulk meta-data migration to DRAM, PACT reduces memory traffic and improves DRAM utilization in SHMs.

Evaluation and results: We evaluate STASH on an SHM system that integrates a 2GB HMC [47] (DRAM cache) with 32GB smart TLC PCM [48]. Both the HMC and the smart TLC PCM are configured to support high-bandwidth SerDes links (120GB/s [20, 47]). Our evaluations use (i) trace-based simulations to study the impact of SHM security on NVM lifetime and write energy, and (ii) system-level simulations to study the impact of SHM security on system performance. We compare STASH with ObfusMem [21] and SHM-SSA. All the evaluated techniques support IDR and integrate SECRET [1] and ASSURE [31] to reduce NVM writes. We refer to low-power ObfusMem and low-power SHM-SSA as $\text{ObfusMem}_{\text{LP}}$ and $\text{SHM-SSA}_{\text{LP}}$, respectively.

Architecture	Memory	Meta-data overhead	IPC	NVM write energy	NVM lifetime
ObfusMem$_{LP}$	Smart NVM	27%	1×	0.52×	2×
SHM-SSA$_{LP}$	SHM	27%	1.32×	0.52×	2×
STASH	SHM	2.1%	1.65×	0.29×	5×

Table 4: **Summary of IPC, NVM write energy, and NVM lifetime results normalized to the baseline.**

Table 4 summarizes the security meta-data overhead, system performance (measured in IPC), NVM write energy, and NVM lifetime results for ObfusMem$_{\mathtt{LP}}$, SHM-SSA$_{\mathtt{LP}}$, and STASH. Results (except meta-data overhead) are normalized to the **baseline** (i.e., ObfusMem without SECRET+ASSURE). We observe that STASH outperforms both ObfusMem$_{\mathtt{LP}}$ and SHM-SSA$_{\mathtt{LP}}$, with the lowest meta-data overhead and NVM write energy, as well as the highest IPC and NVM lifetime.

4. Conclusions and future work

This paper proposed ASSET, which integrates five low overhead, high performance security solutions namely SECRET [1], COVERT [2], ACME [3], ARSENAL [4], and STASH [5] to address the core security challenges of next-generation NVM systems. SECRET ensures low cell flip rate for encrypted MLC/TLC NVMs. COVERT and ACME ensure high system availability of CME by delaying counter overflow. ARSENAL reduces memory write traffic and performance overhead of atomic CME+BMT updates to preserve IDR. Finally, STASH is a comprehensive end-to-end security architecture for state-of-the-art smart hybrid memories. Directions for future research include (i) holistic architectures for both security and reliability of smart hybrid memories, (ii) applications of ASSET to reduce the security overhead of Internet-of-Things, and (iii) secure non-volatile processors, especially in the light of advanced attacks like Spectre and Meltdown.

5. References

[1] S. Swami, J. Rakshit, and K. Mohanram, "SECRET: Smartly encrypted energy efficient non-volatile memories," in *Proc. Design Automation Conference*, 2016.

[2] S. Swami and K. Mohanram, "COVERT: Counter overflow reduction for efficient encryption of non-volatile memories," in *Proc. Design, Automation & Test in Europe Conference & Exhibition*, 2017.

[3] S. Swami and K. Mohanram, "ACME: Advanced counter mode encryption for secure non-volatile memories," in *Proc. Design Automation Conference DAC*, 2018.

[4] S. Swami and K. Mohanram, "ARSENAL: Architecture for Secure Non-Volatile Memories," *IEEE Computer Architecture Letters*, 2018.

[5] S. Swami, J. Rakshit, and K. Mohanram, "STASH: Security architecture for smart hybrid memories," in *Proc. Design Automation Conference*, 2018.

[6] B. C. Lee *et al.*, "Architecting phase change memory as a scalable DRAM alternative," in *Proc. Intl. Symposium on Computer Architecture*, 2009.

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

[8] https://www.micron.com/about/our-innovation/3d-xpoint-technology.

[9] W. Enck *et al.*, "Defending against attacks on main memory persistence," in *Proc. Annual Computer Security Applications Conference*, 2008.

[10] S. Chhabra and Y. Solihin, "i-NVMM: A secure non-volatile main memory system with incremental encryption," in *Proc. Intl. Symposium on Computer Architecture*, 2011.

[11] J. Kong and H. Zhou, "Improving privacy and lifetime of PCM-based main memory," in *Proc. Intl. Conference on Dependable Systems and Networks*, 2010.

[12] V. Young, P. J. Nair, and M. K. Qureshi, "DEUCE: Write-efficient encryption for non-volatile memories," in *Proc. Intl. Conference on Architectural Support for Programming Languages and Operating Systems*, 2015.

[13] 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.

[14] S. Mittal and A. I. Alsalibi, "A survey of techniques for improving security of non-volatile memories," *Journal of Hardware and Systems Security*, 2018.

[15] T. Moscibroda and O. Mutlu, "Memory performance attacks: Denial of memory service in multi-core systems," in *Proc. USENIX Security Symposium*, 2007.

[16] R. B. Lee, "Security basics for computer architects," *Synthesis Lectures on Computer Architecture*, 2013.

[17] C. Yan *et al.*, "Improving cost, performance, and security of memory encryption and authentication," in *Proc. Intl. Symposium on Computer Architecture*, 2006.

[18] 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.

[19] R. Elbaz *et al.*, "Hardware mechanisms for memory authentication: A survey of existing techniques and engines," *Transactions on Computational Science IV*, 2009.

[20] S. Aga and S. Narayanasamy, "InvisiMem: Smart memory defenses for memory bus side channel," in *Proc. Intl. Symposium on Computer Architecture*, 2017.

[21] A. Awad, Y. Wang, D. Shands, and Y. Solihin, "Obfusmem: A low-overhead access obfuscation for trusted memories," in *Proc. Intl. Symposium on Computer Architecture*, 2017.

[22] J. Rakshit and K. Mohanram, "LEO: Low overhead encryption ORAM for non-volatile memories," *IEEE Computer Architecture Letters*, 2018.

[23] 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.

[24] 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.

[25] M. Qureshi *et al.*, "Enhancing lifetime and security of PCM-based main memory with start-gap wear leveling," in *Proc. Intl. Symposium on Microarchitecture*, 2009.

[26] S. Swami and K. Mohanram, "Reliable non-volatile memories: Techniques and measures," *IEEE Design & Test*, 2017.

[27] S. Swami, P. M. Palangappa, and K. Mohanram, "ECS: Error-correcting strings for lifetime improvements in non-volatile memories," *ACM Transactions on Architecture and Code Optimization*, 2017.

[28] C.-K. Luk *et al.*, "Pin: Building customized program analysis tools with dynamic instrumentation," in *Proc. Conference on Programming Language Design and Implementation*, 2005.

[29] A. Patel, F. Afram, and K. Ghose, "Marss-x86: A qemu-based micro-architectural and systems simulator for x86 multicore processors," in *Proc. Design Automation Conference*, 2011.

[30] J. L. Henning, "SPEC CPU2006 benchmark descriptions," in *ACM SIGARCH Computer Architecture News*, 2006.

[31] J. Rakshit and K. Mohanram, "ASSURE: Authentication scheme for secure energy efficient non-volatile memories," in *Proc. Design Automation Conference*, 2017.

[32] J. Lee, T. Kim, and J. Huh, "Reducing the memory bandwidth overheads of hardware security support for multi-core processors," *IEEE Transactions on Computers*, 2016.

[33] Halderman *et al.*, "Lest we remember: Cold-boot attacks on encryption keys," *Communications of the ACM*, 2009.

[34] J. Yang, L. Gao, and Y. Zhang, "Improving memory encryption performance in secure processors," *IEEE Transactions on Computers*, 2005.

[35] T. S. Lehman, A. D. Hilton, and B. C. Lee, "PoisonIvy: Safe speculation for secure memory," in *Proc. Intl. Symposium on Microarchitecture*, 2016.

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

[37] C. Liu *et al.*, "Ghostrider: A hardware-software system for memory trace oblivious computation," *Proc. Intl. Conference on Architectural Support for Programming Languages and Operating System*, 2015.

[38] M. Ekman and P. Stenström, "A robust main-memory compression scheme," in *Intl. Symposium on Computer Architecture*, 2005.

[39] 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.

[40] S. Schechter *et al.*, "Use ECP, not ECC, for hard failures in resistive memories," in *Proc. Intl. Symposium Computer Architecture*, 2010.

[41] M. Qureshi, "Pay-As-You-Go: Low-overhead hard error correction for phase change memories," in *Proc. Intl. Symposium Microarchitecture*, 2011.

[42] P. Rosenfeld *et al.*, "DRAMSim2: A cycle accurate memory system simulator," *IEEE Computer Architecture Letters*, 2011.

[43] M. Qureshi, V. Srinivasan, and J. A. Rivers, "Scalable high performance main memory system using phase-change memory technology," in *Proc. Intl. Symposium on Computer Architecture*, 2009.

[44] S. Swami and K. Mohanram, "E^{3}R: Energy efficient error recovery for multi/triple-level cell non-volatile memories," in *Proc. Intl. Conference VLSI Design*, 2016.

[45] Y. Zhou *et al.*, "HNVM: Hybrid NVM enabled datacenter design and optimization," *Microsoft Research, Tech. Rep.*, 2017.

[46] T. Coughlin, "Crossing the chasm to new solid-state storage architectures," *IEEE Consumer Electronics Magazine*, 2016.

[47] J. T. Pawlowski, "Hybrid Memory Cube (HMC)," in *IEEE Hot Chips Symposium*, 2011.

[48] Stanisavljevic *et al.*, "Demonstration of reliable triple-level-cell (TLC) phase change memory," in *IEEE Intl. Memory Workshop*, 2016.