STASH: SecuriTy Architecture for Smart Hybrid Memories

Shivam Swami Joydeep Rakshit Kartik Mohanram
Department of Electrical and Computer Engineering, University of Pittsburgh, Pittsburgh
shs173@pitt.edu jor73@pitt.edu kartik.mohanram@gmail.com

ABSTRACT

Whereas emerging non-volatile memories (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. Smart hybrid memories (SHMs) that integrate NVM, DRAM, and on-module processor logic are an efficient means to bridge the latency and endurance gaps between NVM-only and DRAM-only memory systems. However, these SHMs are vulnerable to data confidentiality and integrity attacks that can be executed on the unsecure NVM, DRAM, and/or memory buses. **STASH** is the first comprehensive end-to-end SecuriTy Architecture for SHMs that integrates (i) counter mode encryption for data confidentiality, (ii) low overhead page-level Merkle Tree (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 the closest state-of-the-art security solution extensible to SHMs, STASH reduces memory overhead by 12.7×, increases system performance by 65%, and improves NVM lifetime by 2.5×.

CCS Concepts: Security and privacy → Security in hardware; Hardware attacks and countermeasures;

Keywords: Smart Hybrid Memories, Encryption, Authentication

1. Introduction

Whereas novel resistance-class non-volatile memories (NVMs) such as phase change memory (PCM), resistive RAM (RRAM), and 3D XPoint memory [1–3] provide low power, dense, scalable alternatives to DRAM, the high latency and low endurance of these NVMs are impediments to the rapid commercialization of NVM-only memory systems [4–6]. We propose smart hybrid memories (SHMs) as the convergence of the best of hybrid NVM-DRAM memories [4, 5] with smart memories such as the hybrid memory cube (HMC) [7]. In hybrid NVM-DRAM memories, the DRAM caches a majority of accesses to the NVM, effectively concealing the high latency of the NVM and also improving NVM lifetime. In smart memories like the HMC, the integration of on-module processor logic can support high-bandwidth packetized Serializer/Deserializer (SerDes) processor-memory transfers. Thus, SHMs that integrate NVM, DRAM (as the NVM cache), 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 attacks) [8–15] and data integrity attacks (e.g., spoofing, splicing, and replay attacks) [13–16] that must be addressed prior to commercialization.*

This research was supported by NSF Award CCF-1217738.

DAC '18, June 24–29, 2018, San Francisco, CA, USA
© 2018 ACM. ISBN 978-1-4503-5700-5/18/06...$15.00
DOI: https://doi.org/10.1145/3195970.3196123

SHMs pose unique challenges to the direct adoption of state-of-the-art security solutions like ObfusMem [14] and InvisiMem [15], which address data confidentiality, data integrity, and memory side-channel attacks in NVM-/DRAM-only smart memories. These include the (i) presence of both volatile and non-volatile memory modules, which render SHMs vulnerable to security attacks in both operational and powered down states, (ii) need for consistent NVM updates to tolerate power/system failures, i.e., instant data recovery (IDR), and (iii) management and migration of page-granularity data and security meta-data from NVM to DRAM, which increases memory traffic and reduces DRAM utilization. Whereas a strawman solution that addresses these considerations can be derived by extending ObfusMem [14] and InvisiMem [15], this imposes heavy overheads on memory, performance, and NVM lifetime. **STASH** is the first comprehensive end-to-end SecuriTy Architecture for SHMs that makes the following three core contributions.

Low-cost page-level MT authentication, PMT, that replaces classical cache-line-level authentication to secure the SHM from data integrity attacks. PMT leverages page granularity data migration between DRAM and NVM to reduce Merkle Tree (MT) authentication overheads for thwarting data tampering attacks. Bonsai MT (BMT) [16] framework is the norm for memory authentication that is adopted by all state-of-the-art memory security techniques [11, 13, 14]. BMT requires a 128-bit data message authentication code (DMAC) per 512-bit cache line along with an MT constructed over encryption counters (counters henceforth). In contrast, PMT maintains a 128-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.

Recovery-compatible MT updates, RECOUP, is an IDR solution that performs selective meta-data (counters, MT 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 [11, 16]. 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. Extending BMT [16] to support IDR in SHMs 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 trusted computing base (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.

Page-migration-friendly security meta-data management, PACT, supports low overhead page migration is SHMs. Since state-of-the-art security solutions [11, 14–17] 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.

STASH is evaluated on an SHM that integrates a 32GB smart triple-level cell (TLC) PCM [18] as the main memory and a 2GB HMC [7] as the DRAM cache. We used the NVMain memory simulator [19] and the MARSS [20] full system simulator for trace-based and system-level evaluations of STASH, respectively. STASH is compared to (i) state-of-the-art ObfusMem [14] and (ii) a strawman security architecture for SHMs (i.e., SHM-SSA). Results show that in comparison to ObfusMem (SHM-SSA), STASH reduces memory overhead by 12.7× (12.7×), improves system performance by 65% (25%), and increases NVM lifetime by 2.5× (2.5×).

Section 2 provides background on SHM architecture and security. Section 3 describes strawman security architecture for SHM to motivate STASH. Section 4 describes the theory and architecture of STASH. Section 5 presents results and Section 6 is a conclusion.

2. Background

This section presents the SHM architecture and a valid threat model based on state-of-the-art in memory security.

2.1 Smart Hybrid Memories (SHMs)

SHMs integrate smart DRAM [7] with smart NVM [21] to realize high bandwidth, low latency, and high density memory systems. Similar to hybrid DRAM-NVM systems [4, 5, 22], there are two classes of SHMs: (i) hierarchical SHMs (HSHM) with smart DRAM as the cache and a larger smart NVM [4, 22] and (ii) parallel SHMs (PSHM) wherein smart DRAM and smart NVM present a unified interface [5]. Since PSHM requires a large DRAM and changes to the operating system (OS) for memory management, PSHM increases both system cost and complexity. In contrast, HSHM uses DRAM as a small hardware-managed cache without any OS modifications. Figure 1 illustrates both architectures. This work assumes HSHM as the underlying SHM; however the solutions developed here are equally applicable to PSHMs.

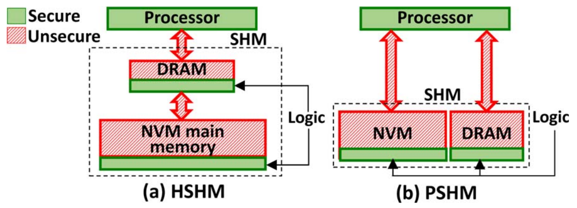

Figure 1: **SHM based on (a) HSHM and (b) PSHM architectures. Whereas HSHM uses a small, OS-transparent smart DRAM as a cache with smart NVM, PSHM uses a large, OS-managed smart DRAM in parallel with smart NVM.**

2.2 Threat Model

It is common practice to design a secure computing system assuming a trusted computing base (TCB) and a set of valid threats, which constitute the threat model of the secure system [9–15]. Along ObfusMem [14] and InvisiMem [15], our TCB consists of the processor chip, the processing logic of the smart DRAM, and the processing logic of the smart NVM, as illustrated in figure 1. Core parts of the OS (e.g., security kernels) are also trusted. Since the TCB is not susceptible to physical attacks, the processor and the processing logic on the smart DRAM/NVM are capable of performing cryptographic operations for SHM security. The memory layers in the smart DRAM and the smart NVM as well as the processor-DRAM and DRAM-NVM memory buses are untrusted. Our threat model encompasses threats to data confidentiality and integrity in both the operational and powered-down system states.

3. Strawman Security Architecture (SSA)

The strawman security architecture for SHMs (SHM-SSA) is a comprehensive end-to-end security framework that also supports instant data recovery in the face of power/system failures.

3.1 Security Primitives of SHM-SSA

For the TCB and threat model summarized in section 2.2, SHMs require protection against: (i) data confidentiality attacks, (ii) data integrity attacks, (iii) side-channel access-pattern-based attacks, and (iv) memory request integrity attacks. The proposed strawman architecture SHM-SSA employs:

1. **Data encryption** for data confidentiality in memory modules and buses: The National Institute of Standards and Technology (NIST) [23] approves that counter mode encryption (CME) based on the advanced encryption standard (AES) block cipher algorithm can provide cryptographic protection for data confidentiality. Along [11, 14, 17], SHM-SSA employs split-counter mode encryption (split-CME) in the processor to encrypt the data stored in main memory. Figure 2(a) shows the encryption/decryption unit (128-bit AES) for data in the core processor.
2. **Memory authentication** for data integrity verification: Without exception, state-of-the-art memory security solutions [11,13,14] use Bonsai Merkle Tree (BMT) [16] for memory authentication. BMT stores a 128-bit data MAC (DMAC) per 512-bit cache line to detect spoofing and splicing attacks, and a hierarchical tree structure (Merkle Tree, i.e., MT) of hashes with the counters as its leaf nodes to detect replay attacks. In SHM-SSA, the DMACs and MT nodes are stored in the smart NVM module and verified on the processor, as shown in figure 2(b).
3. **Access pattern obfuscation** for data confidentiality against side-channel attacks: In SHM-SSA, memory access encryption is used by the sender (processor/memory) and the receiver (memory/processor) for access pattern obfuscation [14,15]. On the processor, SHM-SSA encrypts the memory request, address, and write data before transmission. On the memory, SHM-SSA encrypts the requested read data before transmission. This is shown in figure 2(c). This two-way encryption is possible because the secure logic layers on both smart DRAM and smart NVM are equipped with crypto-engines that employ 128-bit AES CME. These crypto engines maintain separate counters (nonces), but the counters are incremented in lockstep for consistency. This enables OTP pre-computation for encryption/decryption of the next memory access [14, 15]. Note that the ciphertext (from processor/memory) is re-encrypted before transmission to obfuscate temporal re-use of data [14]. Further, to ensure reads and writes are indistinguishable from each other, dummy ciphertext is transmitted by the processor/memory on a read/write request.
4. **Memory bus authentication**, i.e., authenticated encryption for memory request integrity verification: Whereas BMT authentication can detect data tampering in memory module/buses, BMT is ineffective against memory request (command/address) tampering. To detect memory request tampering, memory bus authentication is performed using Galois/Counter Mode (GCM)

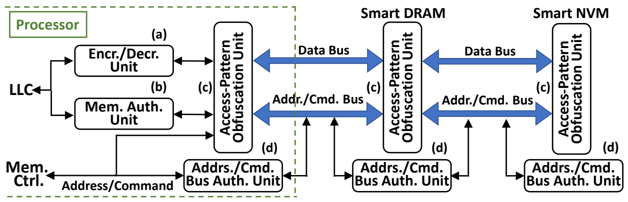
Processor
Smart DRAM
Smart NVM
(a)
Encr./Decr. Unit
(b)
Mem. Auth. Unit
LLC
(c)
Access-Pattern Obfuscation Unit
Data Bus
Addr./Cmd. Bus
(c)
Access-Pattern Obfuscation Unit
Data Bus
Addr./Cmd. Bus
(c)
Access-Pattern Obfuscation Unit
(d)
(d)
(d)
Mem. Ctrl.
Address/Command
Addrs./Cmd. Bus Auth. Unit
Addrs./Cmd. Bus Auth. Unit
Addrs./Cmd. Bus Auth. Unit

Figure 2: **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.**

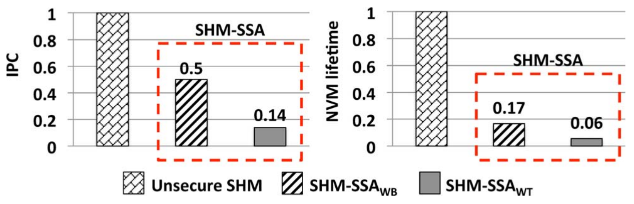

Figure 3: **Comparison of system performance (IPC) and NVM lifetime of unsecure SHM to SHM-SSA$_{\mathrm{WB}}$ and SHM-SSA$_{\mathrm{WT}}$.**

authenticated encryption [15, 24]. In authenticated encryption, the sender (processor/memory) generates an authentication tag (keyed hash value) for the encrypted message (memory access). The tag is also transmitted to the receiver (memory/processor). The receiver regenerates the tag using the received message and verifies it with the received tag to ascertain message integrity. SHM-SSA integrates authenticated encryption logic on the processor as well as memory modules (figure 2(d)).

3.2 SHM-SSA Overheads

The SHM-SSA incurs memory, performance, and endurance overheads over an unsecure SHM architecture, and these are exacerbated by (i) the IDR requirements of SHMs, and (ii) the increased memory traffic and reduced DRAM utilization due to page migration of security meta-data in SHMs.

3.2.1 Security

Figure 3 compares the system performance (measured in instructions per cycle, i.e., IPC) and NVM lifetime of unsecure SHM to SHM-SSA. The SHM integrates a 2GB HMC [7] (as DRAM cache) with a 32GB smart TLC PCM (refer section 5 for framework details). We consider 2 SHM-SSA variants: (i) SHM-SSA$_{\mathrm{WT}}$, which operates smart DRAM and on-chip counter cache in the write-through mode, and (ii) SHM-SSA$_{\mathrm{WB}}$, which operates smart DRAM and on-chip counter cache in conventional write-back mode. Whereas SHM-SSA$_{\mathrm{WB}}$ only ensures security, SHM-SSA$_{\mathrm{WT}}$ ensures both security and IDR (discussed in section 3.2.2). We observe that in comparison to unsecure SHM, SHM-SSA reduces IPC by at least 2× and reduces NVM lifetime by at least 6×. 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 (discussed in section 3.2.3), and (iii) high cell flip rate of the hashes/encrypted data.

3.2.2 Instant Data Recovery

SHMs integrate smart NVMs to enable IDR, which is required for tolerating power/system failures [8, 25], efficient checkpointing [26], and reduced application startup time [27]. Since SHMs use DRAM as a write-back cache to reduce NVM writes, the data residing in the NVM module is stale for the pages that are cached in the DRAM module. In addition, the counters and BMT nodes cached on the processor make reliable data decryption and authentication infeasible during power/system failures, thereby compromising IDR. Whereas SHM-SSA$_{\mathrm{WT}}$ achieves IDR by consistently updating ciphertext and meta-data, it significantly increases NVM write traffic, deteriorating system performance and NVM lifetime.

Figure 3 compares the IPC and NVM lifetime of SHM-SSA$_{\mathrm{WT}}$ with SHM-SSA$_{\mathrm{WB}}$. Results show that SHM-SSA$_{\mathrm{WT}}$ reduces IPC by 3.6× and reduces NVM lifetime by 3× in comparison to SHM-SSA$_{\mathrm{WB}}$. Hence, achieving IDR with a write-through smart DRAM and on-chip counter cache is infeasible in practice.

3.2.3 Page Migration

When SHM-SSA migrates a page from the smart NVM to the smart DRAM, it mandates the migration of its corresponding meta-data to the smart DRAM for data decryption and/or authentication.

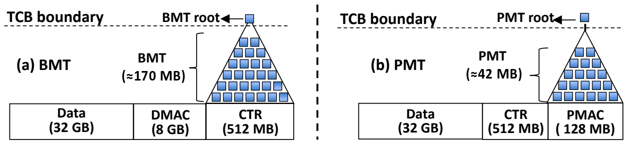

Figure 4: **Data authentication overhead of (a) Bonsai MT (BMT) [16] and (b) page-level MT (PMT). BMT maintains a 128-bit DMAC per 512-bit cache line to detect spoofing and splicing attacks, and a recursively hashed MT over counters to detect replay attacks. In contrast, PMT maintains a 128-bit PMAC per 4kB page to detect spoofing and splicing attacks, and a recursively hashed MT over PMACs to detect replay attacks.**

Since the smart DRAM in SHMs is organized at a 4kB page-level granularity, meta-data migration to the smart DRAM requires migrating multiple 4kB meta-data pages (counter page, DMAC page, etc.), which contain the meta-data corresponding to multiple data pages. This bulk meta-data migration to the smart DRAM is wasteful since the extra meta-data occupies space in the smart DRAM and is not used until the data pages corresponding to this extra meta-data are also migrated to the smart DRAM. Since SHM-SSA uses BMT authentication, which incurs more than 1kB meta-data overhead per 4kB page, the effective DRAM capacity for SHM-SSA is reduced by over 25%.

4. STASH

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.

4.1 STASH: PMT

Page-level Merkle Tree (PMT) is a low overhead authentication solution for SHMs to reduce the memory, performance, and endurance overheads of BMT [16]. In SHMs, data is migrated from the smart NVM to the smart DRAM at 4kB page granularity (i.e., 64 512-bit cache lines), which mandates transferring of 64 DMACs (one for each cache line) for integrity verification. *However, if the 64 DMACs per page are replaced by a single 128-bit page-level MAC (PMAC), the memory and performance overhead of authentication can be significantly reduced without compromising security.*

PMT stores a single 128-bit PMAC per 4kB page to detect spoofing and splicing attacks. Similar to the DMAC, a PMAC is generated by a cryptographic hash function (e.g., NIST-approved SHA-1/2/3) using a secret key, page content, page address, and the corresponding counter. Furthermore, to detect replay attacks, STASH constructs an MT by recursively hashing these PMACs, and stores the root of this page-level MT (PMT) on the trusted logic of the smart DRAM. Since PMT is robust against replay attacks, STASH does not maintain a separate MT over counters.

Figure 4 contrasts memory authentication using BMT and PMT. Without loss of generality, we assume 32GB main memory data is encrypted using the split-counter mode encryption (split-CME [17]), which requires 7-bit minor counter per cache line and 64-bit major counter per 4kB page, i.e., total counter overhead for 32GB data is 512MB. To protect data against spoofing and splicing attacks, BMT requires 128-bit DMAC per 512-bit cache line, i.e., 8GB DMAC for 32GB data. In addition, BMT constructs an MT by recursively hashing the counters (128-bit hash per 512-bit input) and stores the root of the MT on the TCB. The MT overhead for 512MB counters is ≈170MB. In contrast, PMT stores only a single 128-bit PMAC per 4kB page, and constructs an MT over PMACs. The memory overhead of PMACs for 32GB data is 128MB, and the MT overhead is ≈ 42MB. *Thus, PMT reduces the memory overhead of authentication by ≈ 12.7× in comparison to BMT.*

The PMT is stored in the smart NVM and cached in a secure PMAC cache located on the smart DRAM (discussed in section 4.3). The integrity verification of the pages fetched from the smart NVM into the smart DRAM is performed on the processing logic of the smart DRAM. As discussed in [14, 15, 28], a smart-DRAM-like HMC [7] has sufficient area and thermal power budget (55W) to support a low-power processor like the Intel i7-3770T [15, 29]. Hence, PMT caching and processing for integrity verification can be easily performed on an HMC-like smart DRAM [1].

Since PMT performs memory-side data authentication (on the smart DRAM), the data transferred between the smart DRAM and the processor-side memory controller remains vulnerable to integrity attacks. To counter these attacks, STASH also enforces authenticated encryption between the processor and the smart DRAM. However, when a page is evicted from the smart DRAM, it is stored in the smart NVM without any integrity verification. This does not compromise SHM security because the data residing in the smart NVM is always verified by the smart DRAM before it is forwarded to the processor. Hence, even if data tampering is not detected on an NVM write, it is detected by the smart DRAM on the subsequent read of that data. This is consistent with past work [11, 13, 16] on memory authentication, which advocates integrity verification only on read operations.

4.2 STASH: RECOUP

RECOUP addresses the reduced system performance and NVM lifetime limitations of ensuring IDR in SHM-SSA. *RECOUP leverages the key observation that enabling IDR in STASH-based SHMs requires consistent updates to only (a) data (i.e., ciphertext) and its corresponding counter in the NVM module and (b) the PMT root on the TCB of the smart DRAM.* Since STASH generates a PMAC using the page address, data, counter, and a secret key, PMACs can be regenerated after a power/system failure as long as the pages and counters residing in the NVM module are consistently updated. Note that consistent counter updates are also necessary to reliably decrypt the ciphertext after a power/system failure.

Since the PMT is constructed by recursively hashing the PMACs, RECOUP reconstructs the PMT from the regenerated PMACs after a power/system failure. However, to ascertain the authenticity of the recovered data, the root of the regenerated PMT must match the PMT root stored on the TCB of the smart DRAM. Therefore, the PMT root on the TCB of the smart DRAM must be consistently updated on every write. Any non-volatile on-chip storage (e.g., [31,32]) can be used to store the PMT root on the TCB of the smart DRAM, thereby ensuring that the PMT root survives reset and/or power/system failures. Note that consistent PMT root updates do not increase smart NVM writes or impact performance because the PMT root is stored and updated on the smart DRAM TCB.

To consistently update the ciphertext and counters in the smart NVM, STASH uses line-level writes (LLW) [4], which is integral to HSHMs. LLW tracks the writes (at cache-line-level) to the pages residing in the smart DRAM so that only the modified cache lines are written to the smart NVM on page eviction from the smart DRAM. However, instead of waiting for a page eviction, STASH instantly updates a modified cache line and its counter in the NVM module. Note that STASH does not store counters in DRAM (explained in section 4.3), but employs a write-through on-chip counter cache for consistent counter updates.

RECOUP can be supplemented with ciphertext and MAC update reduction techniques like SECRET [12] and ASSURE [13], respectively, to decrease NVM writes. SECRET re-encrypts only the modified non-zero words to reduce NVM writes and employs

[1]Sharing a single DMAC across multiple cache lines was also proposed in [30]; however, since [30] assumes conventional DRAM with processor-side authentication, shared DMAC verification incurs the overhead of fetching the cache lines that share the DMAC to the processor for authentication. In contrast, STASH performs memory-side authentication of the entire page using PMAC without increasing memory traffic.

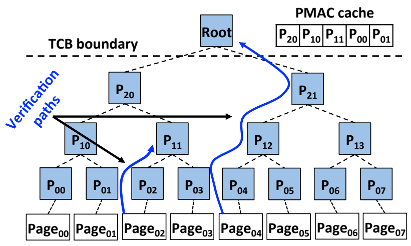

Figure 5: **Illustration of PMT authentication using PMAC cache. The frequently accessed PMT nodes are cached in the PMAC cache and used as local roots to authenticate a fetched page.**

energy masks to reduce the write energy of the ciphertext. ASSURE reduces redundant MAC computations corresponding to the unmodified data and leverages the spatial locality of memory accesses to maintain multi-root MTs that reduce authentication overhead. On integration with PMT, ASSURE maintains a smaller PMT that spans only the hot pages in DRAM and a larger PMT for the rest of the memory. This results in significant latency reduction for the PMT root updates corresponding to the hot pages. Note that integrating SECRET and ASSURE with STASH incurs memory overhead of 6.25% and 1.6%, respectively [12, 13]. Additionally, ASSURE requires a hot/cold PMAC predictor that incurs a logic overhead of 2k 2-input NAND gates [13].

Techniques like DEUCE [10], SECRET [12], and ASSURE [13] are susceptible to side-channel attacks threatening data confidentiality. *However, since STASH employs access-pattern obfuscation and authenticated encryption, the attacker cannot observe address and/or timing of the unmodified data. Thus, integrating SECRET and ASSURE with STASH does not compromise SHM security.*

4.3 STASH: PACT

PACT is a meta-data (PMACs and counters) management solution to reduce the page migration overheads of SHM-SSA. *PACT is motivated by the observation that PMACs and counters exhibit high spatial and temporal locality, i.e., employing cache-based PMAC and counter management strategies reduce memory traffic and improve DRAM utilization.*

PMAC management: PACT reduces PMAC migration and storage overhead by caching frequently accessed PMACs on the processing logic of the smart DRAM. Note that in BMT-based authentication, frequently accessed MT nodes are cached in the on-chip counter cache, requiring a large counter cache to store both counters and MT nodes. Instead of increasing the capacity of the on-chip counter cache, STASH integrates a separate PMAC cache on the TCB of the smart DRAM. The impact of counter and PMAC cache sizes on performance is discussed in section 5.3.

Once a PMT node (including the leaf-level PMACs) is authenticated and stored in the PMAC cache, it is trusted and treated as a local root. As a result, the authentication (on a page fetch) is terminated as soon as a cached PMT node is encountered in the PMAC cache, as illustrated in figure 5. Without loss of generality consider a scenario where smart DRAM requests 2 pages (Page_{02} and Page_{04}) from smart NVM. When a page is fetched from the smart NVM, all the PMT nodes lying on the fetched page's branch are also fetched and verified to detect spoofing, splicing, and replay attacks. However, since the parent hash (P_{11}) of P_{02} is already present in the secure PMAC cache, the authentication process for Page_{02} terminates with the integrity verification of P_{11}. In contrast, since none of the nodes lying on the branch of Page_{04} are present in the PMAC cache, the authentication process for Page_{04} terminates only at the PMT root, which never leaves the TCB. Hence, caching the PMT nodes on the TCB of the smart DRAM not only improves DRAM utilization, but also reduces the authentication overhead (additional hash fetch and verify).

Counter management: Migrating a counter page from the smart NVM to smart DRAM is wasteful, since it transfers extra counters for 63 other pages; these extra counters are not used until their corresponding pages are also migrated to the smart DRAM. PACT eliminates this waste by not storing any counters on the smart DRAM, and instead forwards the required page counters directly to the processor-side write-through counter cache via a pass-thru I/O link [7]. By eliminating bulk counter migration to the smart DRAM, PACT reduces memory traffic and improves DRAM utilization. *Note that to detect counter tampering in smart NVM and data bus, PACT employs PMAC-based counter integrity verification and authenticated encryption, respectively.*

5. Evaluation and Results

We evaluate STASH on an SHM system that integrates a 2GB HMC [7] (DRAM cache) with 32GB smart TLC PCM [18]. Both the HMC and the smart TLC PCM are configured to support high-bandwidth SerDes links (120GB/s [7, 15]). 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.

Evaluated techniques: This section evaluates (i) state-of-the-art ObfusMem [14] (**baseline**), (ii) SHM-SSA (proposed in section 3), and (iii) STASH. All the evaluated techniques support IDR by ensuring consistent ciphertext and meta-data (i.e., counters, hashes, etc.) updates in NVMs. For fairness of comparison, we extend ObfusMem and SHM-SSA to integrate write reduction techniques for secure NVMs, namely SECRET [12] and ASSURE [13]; these are integral to STASH. We refer to low-power ObfusMem and low-power SHM-SSA as ObfusMem$_{LP}$ and SHM-SSA$_{LP}$, respectively.

Cryptographic primitives: The latency of a 45nm fully-pipelined AES core was obtained from [14,15]. We integrate 2 AES cores on the processor and 2 (1) AES cores on the logic layer of the smart DRAM (NVM) module. Whereas ObfusMem [14] provisions a 256kB on-chip counter cache, we employ a 128kB on-chip counter cache and a 128kB PMAC cache on the smart DRAM module.

Trace-driven simulation: For deep, multi-billion instruction evaluation, we perform trace-based simulations using NVMain [19], which is a cycle-accurate main memory simulator. We modified NVMain to reflect the SHM system described above.

Full-system simulation: For system-level simulations, we used the MARSS full-system simulator [20]. MARSS is configured to simulate a standard 4-core out-of-order system running at 3GHz. Each core is assigned a private L1 instruction and data cache of 32kB each and a private L2 cache of 128kB; the L3 cache (LLC here) is a single, shared write-back cache of 8MB. Finally, the main memory is configured to reflect the SHM system described above.

Workloads: To evaluate system performance in real-world scenarios, we use 9 composite workloads (shown in figure 6) with each workload containing 4 SPEC CPU2006 [33] benchmarks.

5.1 Summary

Table 1 summarizes the security meta-data overhead, system performance (measured in instructions per cycle (IPC)), NVM write energy, and NVM lifetime results for ObfusMem$_{LP}$, SHM-SSA$_{LP}$, 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$_{LP}$ and SHM-SSA$_{LP}$, with the lowest meta-data overhead and NVM write energy, as well as the highest IPC and NVM lifetime.

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

5.2 System Performance

Figure 6 illustrates the impact of end-to-end security on system performance (measured in IPC and normalized to the baseline). Along the lines of [11, 16], we employ speculative execution wherein the requested data is forwarded to the processor while integrity verification is performed in the background. Note that on an integrity verification failure a hardware exception is eventually raised by the processor-side memory controller to halt the system and initiate remedial action by the administrator.

Results show that STASH improves IPC by 65% and 25% in comparison to ObfusMem$_{LP}$ and SHM-SSA$_{LP}$, respectively. Note that since we do not assume a power-constrained NVM system, integration of SECRET+ASSURE does not impact IPC [13]. Hence, the IPC of ObfusMem$_{LP}$ (SHM-SSA$_{LP}$) is equivalent to the IPC of ObfusMem (SHM-SSA). Both STASH and SHM-SSA$_{LP}$ register large IPC gains over ObfusMem due to the use of the smart DRAM cache, which reduces the effective read latency of STASH and SHM-SSA$_{LP}$ by $\approx 10\times$ in comparison to PCM-only ObfusMem (42ns for HMC [7, 15] versus 420ns for TLC PCM [18]). Further, STASH also registers high IPC gains over SHM-SSA$_{LP}$ because (i) STASH uses PMACs in place of DMACs and caches frequently accessed PMACs on the TCB of the smart DRAM, resulting in significantly lower page verification and PMT update latency, (ii) RECOUP obviates full MT branch update (in the smart NVM) on every write, reducing the effective write latency, and (iii) PACT improves smart DRAM utilization, resulting in fewer page faults.

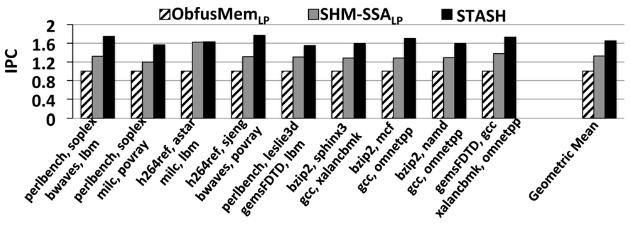

Figure 6: **STASH improves IPC by 65% and 25% in comparison to the ObfusMem$_{LP}$ and SHM-SSA$_{LP}$, respectively, on workloads comprised of SPEC CPU2006 benchmarks [33].**

5.3 Counter Cache and PMAC Cache Sizing

STASH uses 2 types of cache: (a) counter cache and (b) PMAC cache (section 4.1). For fairness of comparison, the combined capacity of the counter and PMAC caches in STASH is equal to that of a single counter cache in ObfusMem/SHM-SSA. This section empirically determines counter cache and PMAC cache sizes that achieve the highest system performance (IPC) for STASH.

Figure 7 reports IPC for 5 different cache sizing schemes. In all cases, the combined capacity of the counter and PMAC cache in STASH is kept equal to the total counter cache capacity of ObfusMe-

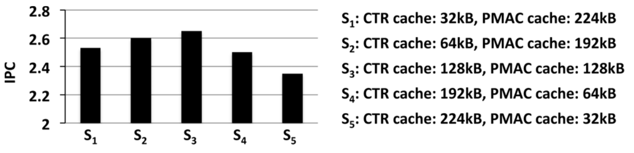

Figure 7: **IPC results for various sizes of counter cache (abbreviated as CTR cache) and PMAC cache. In all cases, the combined capacity of the counter and PMAC caches in STASH is equal to the total counter cache capacity of ObfusMem/SHM-SSA.**

m/SHM-SSA. As shown in the figure, IPC increases steadily until counter cache and PMAC cache are both 128kB in size. When the counter (PMAC) cache is increased (decreased) above (below) 128kB, IPC starts decreasing sharply. This is because the miss rate of the PMAC cache becomes significant below 128kB. Whereas a counter cache miss incurs only a counter fetch and verification penalty, a PMAC cache miss incurs penalties due to: (i) PMAC fetch, (ii) PMAC child verification (which is delayed if the PMAC is not present in the PMAC cache), and (iii) PMAC verification, which propagates all the way to the PMT root if none of the PMT nodes on the fetched PMAC's branch are cached. Hence, equal-size counter and PMAC caches offer the best performance for STASH.

5.4 NVM Write Energy and Lifetime

ObfusMem (by extension) and SHM-SSA/STASH (by design) support IDR, increasing NVM writes (section 4.2) and negatively impacting NVM write energy and lifetime.

Write energy: In Figure 8(a), we observe that the write energy of $\text{ObfusMem}_{\text{LP}}$ and $\text{SHM-SSA}_{\text{LP}}$ are equal because both schemes have been extended to perform consistent ciphertext and meta-data updates (e.g., counters, hashes, etc.) for IDR, resulting in the same NVM write traffic for both schemes. In contrast, STASH employs RECOUP for IDR, which only updates the modified cache line in the NVM module and the PMT root on the secure logic of the smart DRAM, to reduce write traffic and write energy in the smart NVM by 45% in comparison to both $\text{ObfusMem}_{\text{LP}}$ and $\text{SHM-SSA}_{\text{LP}}$.

Further, we also observe that SECRET+ASSURE reduces write energy of $\text{ObfusMem}_{\text{LP}}$, $\text{SHM-SSA}_{\text{LP}}$, and STASH by 48%, 48%, and 71% over baseline ObfusMem that does not integrate SECRET and ASSURE. The impact of SECRET+ASSURE is most evident for benchmarks that consist of a large number of unmodified and/or zero words, e.g., mcf and gobmk.

NVM lifetime: In Figure 8(b), we observe that STASH improves NVM lifetime by 2.5× in comparison to both $\text{ObfusMem}_{\text{LP}}$ and $\text{SHM-SSA}_{\text{LP}}$. The high NVM lifetime of STASH results from reduced cell writes due to RECOUP. Note that integration of SECRET+ASSURE increases the NVM lifetimes of $\text{ObfusMem}_{\text{LP}}$, $\text{SHM-SSA}_{\text{LP}}$, and STASH by 2×, 2×, and 5×, respectively, over baseline ObfusMem.

6. Conclusions

Smart hybrid memories (SHMs) are an efficient means to bridge the latency and endurance gaps between NVM-only and DRAM-only memory systems. STASH is the first comprehensive end-to-end SecuriTy Architecture for SHMs that integrates (i) counter mode encryption for data confidentiality, (ii) low overhead page-level Merkle Tree (MT) authentication for data integrity, (iii) recovery-compatible MT updates to tolerate power/system failures, and (iv) page-migration-friendly security meta-data management. For security guarantees equivalent to state-of-the-art, STASH reduces memory overhead by 12.7×, improves system performance by 65%, and increases NVM lifetime by 5×.

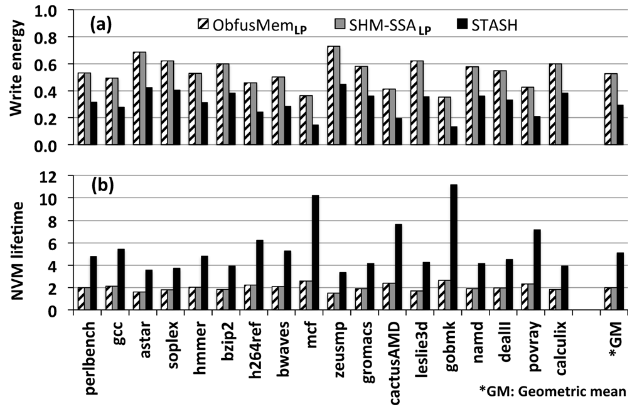

Figure 8: **(a) NVM write energy and (b) NVM lifetime of $\text{ObfusMem}_{\text{LP}}$, $\text{SHM-SSA}_{\text{LP}}$, and STASH, normalized to the baseline (ObfusMem without SECRET+ASSURE). The write energy of STASH is 45% lower than both $\text{ObfusMem}_{\text{LP}}$ and $\text{SHM-SSA}_{\text{LP}}$. Furthermore, NVM lifetime of STASH is 2.5× higher than both $\text{ObfusMem}_{\text{LP}}$ and $\text{SHM-SSA}_{\text{LP}}$. By integrating SECRET+ASSURE, $\text{ObfusMem}_{\text{LP}}$, $\text{SHM-SSA}_{\text{LP}}$, and STASH register significant improvements in write energy and NVM lifetime in comparison to the baseline.**

7. References

[1] B. C. Lee *et al.*, "Architecting phase change memory as a scalable DRAM alternative," in *Proc. ISCA*, 2009.
[2] C. Xu *et al.*, "Understanding the trade-offs in multi-level cell ReRAM memory design," in *Proc. DAC*, 2013.
[3] https://www.micron.com/about/our-innovation/3d-xpoint-technology.
[4] M. K. Qureshi *et al.*, "Scalable high performance main memory system using phase-change memory technology," *Proc. ISCA*, 2009.
[5] G. Dhiman *et al.*, "PDRAM: A hybrid PRAM and DRAM main memory system," in *Proc. DAC*, 2009.
[6] S. Swami and K. Mohanram, "Reliable non-volatile memories: Techniques and measures," *IEEE Design & Test*, 2017.
[7] https://www.micron.com/products/hybrid-memory-cube.
[8] S. Chhabra and Y. Solihin, "i-NVMM: A secure non-volatile main memory system with incremental encryption," in *Proc. ISCA*, 2011.
[9] R. B. Lee, "Security basics for computer architects," *Synthesis Lectures on Computer Architecture*, 2013.
[10] V. Young *et al.*, "DEUCE: Write-efficient encryption for non-volatile memories," in *Proc. ASPLOS*, 2015.
[11] J. Lee *et al.*, "Reducing the memory bandwidth overheads of hardware security support for multi-core processors," *IEEE Trans. on Computers*, 2016.
[12] S. Swami, J. Rakshit, and K. Mohanram, "SECRET: Smartly encrypted energy efficient non-volatile memories," in *Proc. DAC*, 2016.
[13] J. Rakshit and K. Mohanram, "ASSURE: Authentication scheme for secure energy efficient non-volatile memories," in *Proc. DAC*, 2017.
[14] A. Awad *et al.*, "Obfusmem: A low-overhead access obfuscation for trusted memories," in *Proc. ISCA*, 2017.
[15] S. Aga and S. Narayanasamy, "InvisiMem: Smart memory defenses for memory bus side channel," in *Proc. ISCA*, 2017.
[16] 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.
[17] C. Yan *et al.*, "Improving cost, performance, and security of memory encryption and authentication," in *Proc. ISCA*, 2006.
[18] Stanisavljevic *et al.*, "Demonstration of reliable triple-level-cell (TLC) phase change memory," in *IEEE IMW*, 2016.
[19] M. Poremba and Y. Xie, "NVMain: An architectural-level main memory simulator for emerging non-volatile memories," in *Proc. ISVLSI*, 2012.
[20] A. Patel *et al.*, "Marss-x86: A qemu-based micro-architectural and systems simulator for x86 multicore processors," in *Proc. DAC*, 2011.
[21] R. Ramanujan *et al.*, "Dynamic partial power down of memory-side cache in a 2-level memory hierarchy." https://www.google.com/patents/US20140304475, 2014.
[22] Y. Zhou *et al.*, "HNVM: Hybrid NVM enabled datacenter design and optimization," *Microsoft Research, Tech. Rep.*, 2017.
[23] http://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-38a.pdf.
[24] https://csrc.nist.gov/projects/block-cipher-techniques/bcm.
[25] W. Enck *et al.*, "Defending against attacks on main memory persistence," in *Proc. ACSAC*, 2008.
[26] X. Dong *et al.*, "Leveraging 3D PCRAM technologies to reduce checkpoint overhead for future exascale systems," in *Proc. SC*, 2009.
[27] H. Kim *et al.*, "Accelerating application start-up with non-volatile memory in android systems," *IEEE Micro*, 2015.
[28] Y. Eckert *et al.*, "Thermal feasibility of die-stacked processing in memory," in *Proc. Workshop on Near-Data Processing*, 2014.
[29] https://ark.intel.com/products/65525/Intel-Core-i7-3770T-Processor-8M-Cache-up-to-3.70-GHz.
[30] B. Gassend *et al.*, "Caches and hash trees for efficient memory integrity verification," in *Proc. HPCA*, 2003.
[31] X. Wu *et al.*, "Hybrid cache architecture with disparate memory technologies," in *Proc. ISCA*, 2009.
[32] J. Zhao *et al.*, "Kiln: Closing the performance gap between systems with and without persistence support," in *Proc. MICRO*, 2013.
[33] J. L. Henning, "SPEC CPU2006 benchmark descriptions," in *ACM SIGARCH Computer Architecture News*, 2006.