2018 IEEE International Symposium on High Performance Computer Architecture

SYNERGY: Rethinking Secure-Memory Design for Error-Correcting Memories

Gururaj Saileshwar[§], Prashant J. Nair[†*], Prakash Ramrakhyani[‡], Wendy Elsasser[‡], Moinuddin K. Qureshi[§]

[§]*Georgia Institute of Technology* [†]*IBM Research* [‡]*ARM Research*

{gururaj.s,moin}@gatech.edu Prashant.Nair.J@ibm.com {Prakash.Ramrakhyani,Wendy.Elsasser}@arm.com

***Abstract*—Building trusted data-centers requires resilient memories which are protected from both adversarial attacks and errors. Unfortunately, the state-of-the-art memory security solutions incur considerable performance overheads due to accesses for security metadata like Message Authentication Codes (MACs). At the same time, commercial secure memory solutions tend to be designed oblivious to the presence of memory reliability mechanisms (such as ECC-DIMMs), that provide tolerance to memory errors. Fortunately, ECC-DIMMs possess an additional chip for providing error correction codes (ECC), that is accessed in parallel with data, which can be harnessed for security optimizations. If we can re-purpose the ECC-chip to store some metadata useful for security and reliability, it can prove beneficial to both.**

To this end, this paper proposes *Synergy*, a reliability-security co-design that improves performance of secure execution while providing strong reliability for systems with 9-chip ECC-DIMMs. Synergy uses the insight that MACs being capable of detecting data tampering are also useful for detecting memory errors. Therefore, MACs are best suited for being placed inside the ECC chip, to be accessed in parallel with each data access. By co-locating MAC and Data, Synergy is able to avoid a separate memory access for MAC and thereby reduce the overall memory traffic for secure memory systems. Furthermore, Synergy is able to tolerate 1 chip failure out of 9 chips by using a parity that is constructed over 9 chips (8 Data and 1 MAC), which is used for reconstructing the data of the failed chip. For memory intensive workloads, Synergy provides a speedup of 20% and reduces system Energy Delay Product by 31% compared to a secure memory baseline with ECC-DIMMs. At the same time, Synergy increases reliability by 185x compared to ECC-DIMMs that provide Single-Error Correction, Double-Error Detection (SECDED) capability. Synergy uses commercial ECC-DIMMs and does not incur any additional hardware overheads or reduction of security.

***Keywords*-Memory Security; Reliability; ECC-DIMMs**

I. Introduction

Memory security is a major concern today. Several recent studies [1], [2], [3], [4], [5] have exposed vulnerabilities that allow an adversary unauthorized access to sensitive regions of the memory, given physical access to the system. These vulnerabilities can be exploited by the adversary to gain control of the system. Furthermore, the proliferation of cloud computing and remote data-centers has made these security concerns more important. To address some of these issues, commercial products like Intel Software Guard Extensions (SGX) provide security for regions of the main memory [6], [7]. We envision that such mechanisms will be extended for securing the entire memory for resilient memory systems.

Providing security for commodity off-chip memory systems results in significant performance overheads. This is because secure memory systems are required to store and provide metadata via additional off-chip accesses. These security metadata include counters, message authentication codes (MACs), and an integrity tree that is traversed on each access. We observe that accessing this metadata increases the off-chip memory traffic and causes upto 60% performance reduction. If we can fetch some of the metadata concurrently while accessing data, we can reduce the memory traffic and improve performance. To this end, this paper investigates secure memory organizations that can provide security metadata with low bandwidth overheads.

In addition to providing security, a resilient memory system must be robust to naturally occurring failures. Several field studies [8], [9] have highlighted the reliability problems that plague DRAM chips. To make matters worse, DRAM chips are projected to encounter higher rates of failure as they scale. Systems requiring high reliability predominantly use memory modules equipped with extra chips to support ECC (ECC chips). However, commercial memory systems tend to be designed for reliability while being oblivious to security mechanisms. Therefore, such secure memory designs do not exploit ECC chips optimally.

This paper advocates re-purposing ECC chips to store security metadata. As ECC chips are accessed in parallel with data, such an organization can reduce the number of metadata accesses. Ideally, we would like to store a metadata that benefits both security and reliability of the memory system. MACs help the security apparatus detect data tampering as they are cryptographic signatures of data. At the same time, MACs can detect memory failures as they result in data corruption [10]. Therefore, MACs are an ideal choice for the metadata to be placed in ECC chips.

Leveraging these insights, we propose *Synergy*, a co-design of reliability and security for systems using ECC-Dual In-line Memory Modules (ECC-DIMMs). Synergy not only improves the performance of secure memory systems, but also provides strong memory reliability. The key insight behind Synergy is the co-location of MAC and data. Commercial x8 ECC-DIMMs tend to have 8 data chips and one additional ECC-chip. Synergy stores the MAC in the ECC-chip (9th chip) thereby enabling it to use the additional bandwidth of the ECC-chip to provide the MAC in the same access as data as shown in Figure 1(c). This avoids the requirement for a separate memory access for MAC on each data access as shown in Figure 1(a) and reduces memory traffic when compared to conventional designs.

*This work was performed when Prashant J. Nair was affiliated with Georgia Institute of Technology.

DOI 10.1109/HPCA.2018.00046

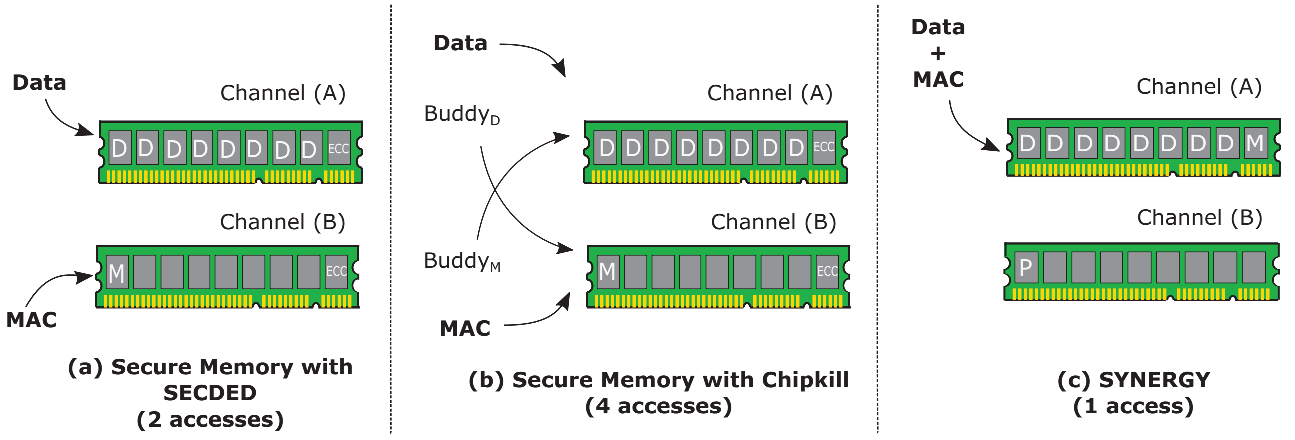


Figure 1: Synergy reduces memory accesses for security metadata MACs. (a) Secure memory with SECDED requires separate memory accesses for data and MAC (b) Secure memory with Chipkill needs dual-channel operation, thereby doubling the number of memory accesses (c) Synergy co-locates data and MAC, thereby providing both in a single memory access.

Synergy re-uses MACs for error detection by exploiting their ability to detect data modifications (with a very high probability). In the rare event of a faulty chip as detected by the MAC, an 8-byte Parity constructed over the contents of the 9 chips (8 data chips and 1 MAC chip) is used to correct errors. This mechanism allows Synergy to tolerate any chip failure within a group of 9 chips (ECC-DIMM).

Typically the 9-chip ECC-DIMMs use Single Error Correction Double Error Detection (SECDED) codes which can correct only single-bit-errors. By tolerating entire chip failures, Synergy provides much stronger reliability than secure memory designs that use SECDED-based ECC-DIMMs. Traditional memory systems use costly techniques like Chipkill [11] to achieve a similar level of reliability. To fix a single chip failure, Chipkill requires accessing 18 chips (two x8 ECC-DIMMs) typically across two channels as shown in Figure1(b) [12], [13]. As 2x more channels are occupied on every access, Chipkill naturally reduces channel-level parallelism and reduces memory bandwidth, thereby causing a slowdown. On the other hand, Synergy provides Chipkill-level reliability by accessing only a single channel and unlocking higher channel-level parallelism. Furthermore, Synergy improves the performance of secure memory systems by co-locating MAC and data.

We compare Synergy to a baseline system that uses ECC-DIMMs for providing SECDED reliability. Our evaluations with 29 memory intensive SPEC2006 and graph workloads show that Synergy improves performance by 20% and reduces the System Energy-Delay-Product by 31%. By providing Chipkill-level reliability by using only a single ECC-DIMM, Synergy reduces the probability of system failure by 185x as compared to the baseline system. Synergy achieves these benefits without any additional hardware or reduction in security. We also compare Synergy to prior work that follows similar spirit of security and reliability co-design. Our comparisons against IVEC [10], a design that combines security and reliability for commodity (non-ECC) DIMMs, show that Synergy provides 60% higher performance than IVEC, while providing similar reliability to chip failures.

II. Background and Motivation

A. Securing Commodity-DRAM

1) Attack Model: Secure memories provide protection from adversaries with physical access to the system. Similar to past works [14], [15], our attack model assumes the processor as the trusted computing base. All off-chip resources including the memory bus and physical memory are vulnerable to unauthorized reads or modifications by attackers. Additionally, an adversary may perform a man-in-the-middle attack by intercepting the memory bus and replaying stale data values. In such a setting, securing commodity memories involves ensuring data confidentiality with counter-mode encryption, data integrity with MACs, and replay attack protection using integrity-trees. These security metadata are fetched from memory through additional memory accesses, on each off-chip memory access for program data.

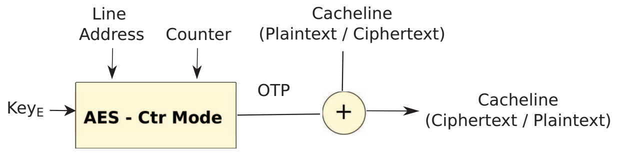

Line Address
Counter
Cacheline (Plaintext / Ciphertext)
Key$_E$
AES - Ctr Mode
OTP
+
Cacheline (Ciphertext / Plaintext)

Figure 2: Counter-Mode Encryption

2) Confidentiality with Counter-Mode Encryption: To prevent attackers from reading sensitive memory contents, data is stored in memory using Counter Mode Encryption [16], [17]. As shown in Figure 2, during encryption, a cipher-text is generated from a plain-text cacheline through an XOR with a One Time Pad (OTP). Decryption is performed by an XOR of the cipher-text cacheline with the same OTP. The OTP is a secret bit-string unknown to the adversary generated using a block cipher like Advanced Encryption Standard (AES) and a secret key only known to the processor. A per-line counter, that is incremented on each cacheline-write, is used as an input to AES to ensure temporal variation in the encrypted data. To avoid fetching the counter from memory on each data access and allow for pre-computation of the OTP, the counter is cached on-chip in a dedicated cache [6] or in the last-level cache [18].

3) Integrity using Message Authentication Codes: To prevent unauthorized modification of memory contents, cryptographic signatures of cacheline contents called message authentication codes (MACs) are stored in memory per data-cacheline. As shown in Figure 3, MACs are created using a cryptographic hash function (e.g. 64-bit AES-GCM based GMAC [17]) with a secret key known only to the processor. On every cacheline access, the memory controller recomputes the MAC using the cacheline contents and the corresponding counter and verifies it with the stored MAC. Any modification to the cacheline results in a MAC mismatch, causing the memory controller to declare an attack.

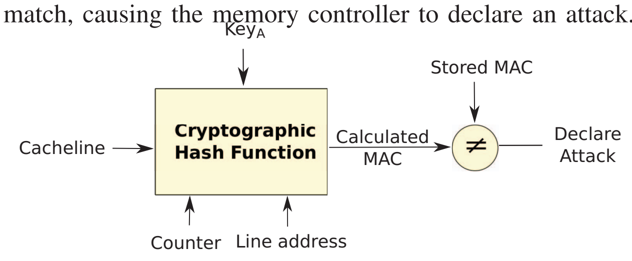


Figure 3: Integrity Verification using MACs

4) Replay Attack Protection: An attacker with access to the bus between the on-chip microprocessor and the off-chip memory could replace an entire tuple of {*Data, MAC, Counter*} with an older copy without detection. To prevent such replay attacks, secure memories use an integrity tree [6], [14], [19], a data-structure for ensuring integrity of a large area of memory with a limited on-chip storage.

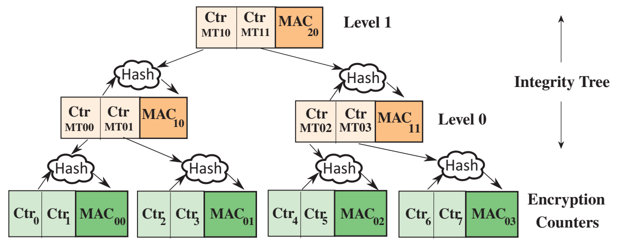


Figure 4: Integrity Tree for Replay Attack Protection

Figure 4 shows a 2-ary counter tree that protects the encryption counters from replay. At each level, the counters are equipped with a MAC for integrity, generated using a higher level tree-counter (e.g., MAC_{00} for the contents of Ctr_0 and Ctr_1 is generated using Ctr_{MT00} in Level-0). On each memory access to the encryption counter (Ctr_0), it's integrity can be verified by accessing the Level-0 counter (Ctr_{MT00}) and computing the MAC (MAC_{00}). The integrity of the Level-0 counter can in-turn be verified by accessing the Level-1 counter and so on. As the number of counters exponentially reduces at each level, the root is small enough to be always stored on-chip, secure from tampering. Thus, a replay of an encryption counter can be prevented by traversing the integrity tree and verifying the integrity of each level, until the trusted root is reached. While the counter tree in Figure 4 only provides replay protection for encryption counters in a Bonsai-style [14], that is sufficient to prevent replay of the {*Data, MAC, Counter*} tuple.

Alternately, an integrity tree can be a tree of hashes, i.e. MAC tree or Merkle tree [20], where each level is a series of hashes based on its lower level. Typically, 8-ary counter trees [6] or MAC trees [17] have been used in prior work.

5) Memory Security in Intel SGX: Intel Software Guard Extensions (SGX) is a commercial product that provides memory security [6] for enclaves (small regions) in memory. It uses 56-bit counters for encryption and integrity, 56-bit MACs based on Carter-Wegman MAC design [21] and an 8-ary counter tree based integrity tree. Additionally, the counters (encryption and integrity-tree) are cached in a dedicated cache on-chip. In this paper, we use an SGX-like design extended to secure the entire memory. In addition, we use an optimized baseline (SGX_O) that caches counters in the last-level cache [18] in addition to the dedicated counter-cache. For consistency across designs, we assume 64-bit GMAC based MACs for SGX and SGX_O.

B. Strong Reliability in Secure Memory

In large-scale production grade clusters, memory related errors have been reported as the leading cause of hardware crashes [9], [22]. In the absence of any error correction mechanism, a secure memory would flag memory errors as an attack, as it has no means of distinguishing between modifications to data due to errors and attacks. To prevent such false positives, a secure memory system must be equipped to tolerate memory errors.

Table I. DRAM failures per billion hours (FIT) [8]

DRAM Chip Failure Mode	Fault Rate (FIT)	
	Transient	Permanent
Single bit	14.2	18.6
Single word	1.4	0.3
Single column	1.4	5.6
Single row	0.2	8.2
Single bank	0.8	10
Multi-bank	0.3	1.4
Multi-rank	0.9	2.8

Table I shows real-world failure probabilities for DRAM devices, as per Sridharan et.al [8]. Secure memory systems can use ECC-DIMMs that store single-error-correction, double-error-detection (SECDED) [23], [24] codes to tolerate single-bit errors. Since they make up 50% of the failures, SECDED provides a 2x improvement in probability of system failure. ECC-DIMMs provide similar performance as Commodity DIMMs, as they provision additional storage and memory bus-width for SECDED codes. For protection against multi-bit failures (failing rows, banks etc.), which occur as often as single-bit failures, solutions like Chipkill [11] use symbol-based error correction [25]. They provide the ability to correct errors arising from the failure of one entire DRAM chip out of 18 chips, reducing system failure probability by 42x compared to a system with SECDED protection. However, Chipkill incurs a performance slowdown

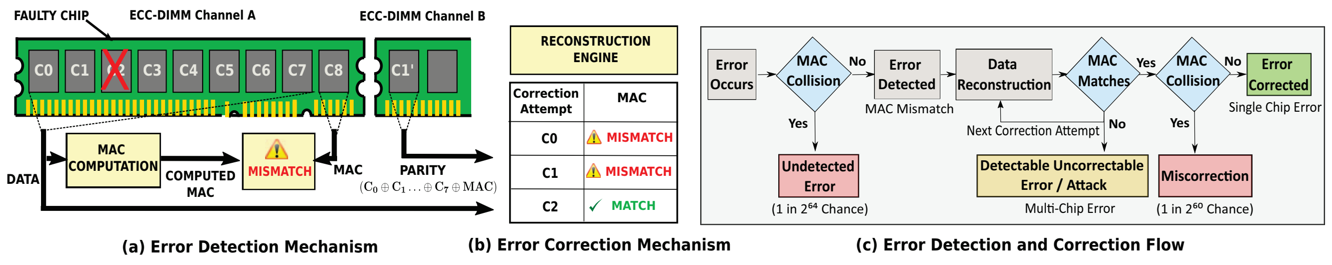


Figure 5: Synergy provides Chipkill-level reliability using MAC for Error Detection and 9-Chip Parity for Correction.[1]

when implemented with x8-DIMMs owing to over-fetch of cachelines [13] and loss of channel-level parallelism [12] due to its dual-channel lock-step operation.

C. Performance Problem in Secure Memory

Securing commodity DRAM requires storing security metadata in memory and accessing it on each off-chip memory access for program data. This leads to performance slowdown during secure execution, especially for memory-intensive workloads requiring frequent accesses to memory.

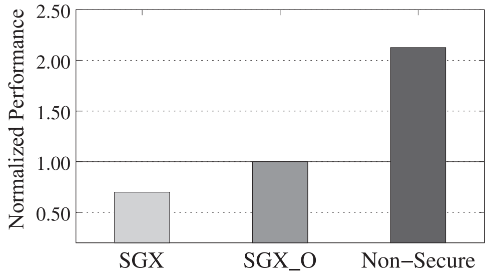


Figure 6: Performance of SGX, SGX_O and Non-Secure, all normalized to SGX_O.

Figure 6 compares performance of secure execution (SGX and SGX_O) against non-secure, for memory intensive workloads from SPEC2006 and GAP [26]. Non-Secure is 112% faster than optimized baseline SGX_O that caches counters in last-level cache. SGX is 30% slower than SGX_O as it caches counters in a small dedicated cache. Thus, there is a considerable performance gap between non-secure and secure execution that needs to be bridged.

D. Insight: Security-Reliability Co-design

Modern server systems largely use memory modules provisioned with additional storage for error correction (ECC-DIMMs), as such systems have an indispensable need for reliability. Current-generation x8 ECC-DIMMs possess an additional ECC-chip providing 12.5% additional storage and bandwidth dedicated for error correction codes. Additionally, secure memories use MACs for data integrity, that can also act as strong error detection codes given their ability to detect data tampering with a high probability [10]. Using MACs for error detection can make the ECC fetched from the ECC-Chip redundant in common-case error-free accesses. This latent ECC-Chip bandwidth may be used towards improving the performance of secure execution. The goal of this paper is to combine memory security and reliability using this insight and not only improve performance but also provide better reliability.

III. Synergy Design

Synergy enables a symbiotic relationship between reliability and security, when used with a 9-chip ECC-DIMM. By storing MAC (8-byte GMAC) in the 9th Chip of the ECC-DIMM as shown in Figure 5, Synergy is able to co-locate MAC and data. As this allows MAC to be fetched in the same access as data avoiding a separate access for MAC, Synergy reduces the overall application memory traffic and improves the performance of secure execution.

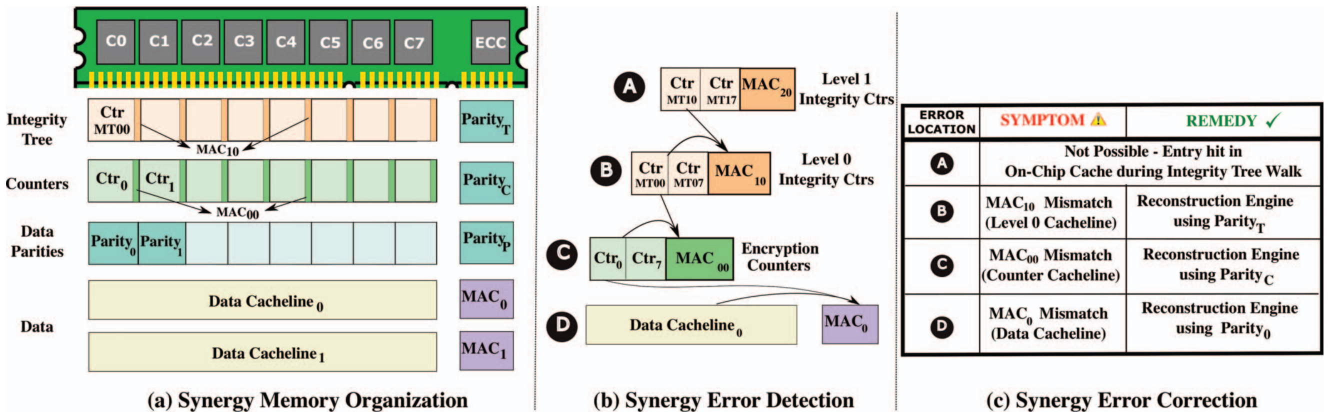
In addition, Synergy also improves reliability by providing tolerance to chip-level failures, compared to SGX_O with an ECC-DIMM storing SECDED code, that is only capable of tolerating single-bit errors. As shown in Figure 5(a), Synergy re-uses the MAC for error detection. In the event of an error due to a faulty chip, there is a mismatch between the computed MAC and the stored MAC. In this case, an 8-byte Parity ($C_0 \oplus C_1 \oplus C_2 \cdots \oplus C_7 \oplus MAC$) constructed over 9-Chips (8 data chips and 1 MAC chip) is used to reconstruct the data (e.g. $C_2 = Parity \oplus C_0 \oplus C_1 \oplus C_3 \cdots \oplus C_7 \oplus MAC$). Since the identity of the erroneous chip is unknown, the RAID-3 based Reconstruction Engine sequentially checks for an error in each of the 9 chips, reconstructing the data and verifying if the re-computed MAC matches, as shown in Figure 5(b). As long as the error only impacts one chip out of a set of 9 chips (8 Data, 1 MAC), Synergy is able to reconstruct the entire data of the failed chip (assuming the Parity is non-erroneous). Thus, Synergy can correct 1 chip failure out of 9 chips and provide stronger reliability than conventional Chipkill, that can only tolerate 1 chip failure out of 18 chips.

Note that while the reconstruction is liable to suffer miscorrection because of a MAC-collision, the probability of this event is negligible (2^{-61} for 8 MAC re-computations). Additionally, because the value corrected by the Parity is always verified with the MAC, Synergy maintains the security of the underlying secure memory system. The subsequent section describes how the error detection and correction algorithm integrates with the integrity tree traversal, and the handling of corner cases like error in counters or parities.

A. Memory Organization

Figure 7(a) shows the memory organization of Synergy, implemented with a 9-chip ECC-DIMM. There exist four types of cachelines, where a potential error might occur:

[1]Although Fig 5 depicts parity and data in separate channels, Synergy design can accommodate both in the same channel as well.



ERROR LOCATION	SYMPTOM ⚠	REMEDY ✓
A	Not Possible - Entry hit in On-Chip Cache during Integrity Tree Walk	
B	MAC_{10} Mismatch (Level 0 Cacheline)	Reconstruction Engine using $Parity_T$
C	MAC_{00} Mismatch (Counter Cacheline)	Reconstruction Engine using $Parity_C$
D	MAC_0 Mismatch (Data Cacheline)	Reconstruction Engine using $Parity_0$

Figure 7: Synergy design for improving performance by co-locating MAC with data, while providing Chipkill-level reliability (a) Memory Organization (b) Potential Error Scenarios - Detection (c) Potential Error Scenarios - Correction

Data cacheline: Synergy co-locates data and MAC, with the 64-byte data stored in 8 chips and the 8-byte MAC stored in the 9th-ECC-chip. An error in either the data or the MAC results in a mismatch during the MAC computations for data integrity. E.g., In Figure 7(b), an error in Data Cacheline$_0$ or MAC$_0$ causes a mismatch in the MAC$_0$ computation.

Parity cacheline: Since MACs are stored in the ECC-chip, the parities are stored in a separate area in memory. Synergy stores eight 8-byte parities in a cacheline, such that each chip C_i supplies a single parity as shown in Figure 7(a). An error in parity is critical only if the data cacheline it protects is erroneous at the same time (e.g., a failed chip contained Data and its Parity in separate cachelines). To tolerate such scenarios, the ECC-chip stores a parity of parities ($Parity_P = P_0 \oplus P_1 \oplus P_2 \cdots \oplus P_7$) for the parity cacheline. We cover the scenario of erroneous parity while addressing error correction for data cachelines.

Encryption counter cacheline: The counter cacheline structure in Synergy is similar to SGX and contains eight 56-bit counters and one 64-bit MAC organized such that each chip provides a single 56-bit counter and 8-bits of the MAC. As a result, a chip failure causes a single counter in the cacheline to become erroneous along with a portion of the MAC provided by the chip. This leads to a mismatch during the MAC computation for the data cacheline (MAC$_0$ in Figure 7(b)) using the erroneous counter (Ctr_0) and a mismatch in the MAC computed during the counter tree walk (MAC$_{00}$). To provide reliability in such scenarios, Synergy stores 8-byte parities ($Parity_C = C_0 \oplus C_1 \oplus C_2 \cdots \oplus C_7$) constructed over 8 chips, in the ECC-chip for these cachelines.

Integrity tree counter cacheline: The integrity tree counter cacheline is similar in structure to an encryption counter cacheline. Each cacheline has eight 56-bit Integrity Tree Counters (Ctr_{MT}) and one 64-bit MAC, with each chip providing a single Ctr_{MT} and 8-bits of MAC. Additionally, an 8-byte parity ($Parity_T$) is stored in the ECC-chip.

Thus for error-handling purposes, there are two types of cachelines - Data cachelines and Counter cachelines.

B. Error Detection and Correction

Figure 7(b) shows the operations performed as a part of integrity-tree traversal, on a data-cacheline read. For example, reading Data Cacheline$_0$ from memory requires a MAC computation (MAC$_0$) and a counter tree traversal involving a series of MAC computations (MAC$_{00}$, MAC$_{10}$, MAC$_{20}$). The tree traversal continues until a particular entry hits in the on-chip cache (e.g. Level 1 entry). A memory error can manifest as a mismatch in any of these MAC computations. However, it is difficult to pin-point the cause of a particular MAC mismatch as there are multiple potential sources of errors (e.g. MAC$_0$ mismatch can be due to an error in MAC$_0$, Ctr$_0$ or Data$_0$). To solve this problem, Synergy uses an error detection and correction algorithm that is integrated with the integrity tree traversal.

Upward tree traversal for integrity / error detection: On every memory access, Synergy traverses the integrity tree from the bottom to the top, as shown in Figure 7(b) to protect against a replay attack. On any MAC mismatch, rather than declaring an attack immediately, Synergy continues the walk logging any mismatches as they occur. This process ends when the tree-walk finds a Ctr_{MT} entry cached on-chip. This entry is assumed to be free from errors since it is found on-chip. For example, Level 1 Ctr_{MT} (shown as A in Figure 7(b)) is found on-chip and does not require any error correction (Scenario A in Figure 7(c)).

Downward tree traversal for error correction: Next, the tree-traversal in the downward direction is performed for error correction, from top to the bottom as shown in Figure 7(b). At each level, forward progress is made if errors at the previous level (higher in the tree) are corrected and mismatches are cleared by using the parity to reconstruct the data as shown in Figure 5. Therefore, at each level in case of a mismatch, the root-cause could only be errors in the same cacheline as the MAC. For example, in Figure 7(b), a mismatch at B could only mean an error in the Level 0 integrity tree cacheline as the other input to MAC$_{10}$ is from a higher level in the tree (Level 1 Ctr_{MT}) and must be correct since it was visited earlier by the algorithm. At any level, if the error is not correctable, i.e. parity is unable to convert the mismatch into a match, an attack is declared ensuring the security of the memory contents.

As shown in Figure 7(c), there are two broad error correction scenarios:

Error correction for counter cachelines: A mismatch of the MAC in the counter or integrity tree cacheline is possible only due to an error in the same cacheline. It can be corrected using a parity-based data reconstruction technique similar to Figure 5. Synergy tries to reconstruct the contents of each of the 8 chips (C_0 to C_7) using the parity ($Parity_C$ or $Parity_T$ as applicable). A match of the stored MAC in the cacheline with the recomputed MAC indicates the success of each reconstruction attempt (Scenarios Ⓑ and Ⓒ in Figure 7(c)). In this case, the reconstruction engine makes a maximum of 8 MAC re-computations per cacheline.

Error correction for data cachelines: Mismatch in MAC computation for the Data cacheline could be due to an error in any of the 9 chips (8 data and 1 MAC). Synergy sequentially tries to reconstruct the contents of each of the 9 chips using the parity (e.g. P_0 as shown in Scenario Ⓓ in Figure 7(c)) and attempts to resolve the MAC mismatch. First, Synergy attempts reconstruction of the MAC chip contents. If unsuccessful, Synergy sequentially attempts reconstruction of other 8 data chips.

There is a chance that all the reconstruction attempts are unsuccessful due to an erroneous parity. This is possible when the Parity ends up in the same faulty chip as the data, in separate cachelines. In this scenario, the parity of parities ($Parity_P$) stored in the ECC-Chip along-with the parity cacheline, is used to reconstruct the erroneous parity. The data reconstruction in this case is attempted with both the original parity and the reconstructed parity. Thus, up to 16 MAC re-computations may be required to correct an erroneous data cacheline.

Detected Uncorrectable Errors or Attack: Synergy's error correction mechanism optimistically attempts to correct errors under the assumption that no more than one out of 9 chips is erroneous at any given time. However, it is possible that more than one chip suffers from errors at the same time - although failure probabilities from Table I indicate that this would be a rare occurrence. Furthermore, an adversary could modify data in multiple chips at the same time. In such scenarios, while the MAC would detect the modification to data, the reconstruction engine would not be able to correct it. In this scenario, Synergy declares an attack since it is unable to identify the source of a data modification - a naturally occurring error or a malicious attack. Thus any malicious modification that cannot be corrected is flagged as an attack, retaining the security of the system.

IV. Discussion

A. Implications for Reliability

Reliability to Chip Failures: For counter cachelines, Synergy uses the chip-level parity stored in the same line constructed over 8 data chips to correct the counters in those 8 chips. For Data+MAC cachelines, the parity is constructed over 9 chips (8 data and 1 MAC chip) and stored in a separate cacheline (potentially in a chip on a different channel). Because the $Parity_P$ is available to fix errors in the parity cacheline, Synergy is still able to correct an overlapping error between any of 9 Data+MAC chips and the Parity chip. As a result, in all cases, Synergy is able to guarantee correction of all-bit errors as long they are restricted to 1 chip out of a set of 9 chips.

Probability of Mis-correction or Silent Data Corruption: To correct each MAC mismatch, Synergy performs a maximum of 16 MAC computations (in the case that data and parity are simultaneously in error). Mis-correction of data by the Reconstruction Engine can occur only if it encounters a hash conflict during the sixteen reconstruction attempts (event probability less than 10^{-20}). Given that reconstruction is triggered only on an error (assuming a conservative rate of 100 failures per billion hours[2]), the overall Silent Data Corruption (SDC) rate of Synergy is less than once per 10^{14} billion years (or an SDC Failure In Time (FIT) rate of 10^{-19}), about thirteen orders of magnitude lower than Chipkill-based non-secure memory systems.

Mitigating Correction Latency under Permanent Chip Failures: Synergy can require up to 88 MAC computations [3] to correct errors from a single failed chip, depending on the number of MAC mismatches on each data access. This can result in an unacceptably large correction latency in case of a permanent chip-failure causing frequent errors. A simple way to mitigate this is to track the faulty chip for each error corrected. After a sufficiently large number of errors have been detected reporting the same faulty chip, Synergy can preemptively correct the identified chip's data with the parity on each access before verifying the integrity with a MAC computation. Thus the correction overhead on a chip failure is reduced to 1 MAC computation, that is anyway required in the baseline secure memory design for data integrity.

Storage Overheads: Synergy incurs a 12.5% storage overhead for storing the Parity (8 bytes per 64-byte cacheline), which is required for error correction. However, this is equivalent to the 12.5% storage overhead for SECDED-ECC in the baseline design with SGX using ECC-DIMMs (8 byte ECC per 64-byte cacheline) or an SGX design with Chipkill (2 ECC symbols per 16 data symbols). The reliability storage overhead in each of these designs is in addition to the storage overheads for security metadata (encryption counters - 12.5%, MACs - 12.5% and integrity tree - 1.8%).

[2]Assumed rate of DRAM transient failures - 20 per billion hours. [8]
[3]For a 9-level Integrity Tree protecting a 16GB memory.

Design	Integrity Tree	Counter		MAC		Reliability Design
		Design	Caching	Design	Caching	
SGX [6]	Bonsai Counter-Tree	Monolithic (56-bit)	Dedicated Cache	64-bit GMAC	None	SECDED
SGX_O	Bonsai Counter-Tree	Monolithic (56-bit)	Dedicated and LLC	64-bit GMAC	None	SECDED
Synergy	Bonsai Counter-Tree	Monolithic (56-bit)	Dedicated and LLC	64-bit GMAC	None	MAC+Parity Co-design
IVEC [10]	non-Bonsai GMAC-Tree	Split (64-bit Major, 7-bit Minor)	Dedicated Cache	64-bit GMAC	LLC	MAC+Parity Co-design

Table II. Secure Memory Designs Evaluated

B. Implications for Security

Impact of repeated MAC computations: Successive MAC computations for verifying error correction success reduces the effective strength of the MAC. The reconstruction engine performs at-most 16 MAC re-computations for a particular 64-bit MAC in the Data Cacheline, reducing the effective strength of MAC to 60-bits. For a MAC in the counter cacheline, the reconstruction engine performs at-most 8 MAC recomputations, reducing the effective strength of the MAC to 62-bits. However, this trade-off is acceptable since Synergy still provides stronger protection than commercial SGX, which only uses a 56-bit MAC [6].

Parity tampering by adversary: As parity is stored unprotected in the memory, it can be modified by an adversary. However, a tampered parity may only be used in the event of a MAC mismatch. In the common case, a tampered parity will be unable to correct the error behind the mismatch and cause Synergy to declare an attack, retaining the security of the system. The probability of a tampered parity causing data mis-correction is negligible as this is equivalent to the attacker committing MAC forgery (probability $\ll 10^{-20}$).

Vulnerability to denial of service attacks: An adversary might modify memory expressly with the intent of creating correctable errors, so that Synergy incurs the latency of multiple MAC re-computations leading to a denial of service. While this does not affect correctness, it impacts performance. To allow for detection of such scenarios, the memory controller may log all incidents of corrected memory errors, with statistical analysis used to distinguish artificially created errors from naturally occurring ones.

Resilience to bit-flip attacks: Attacks like Row Hammer [3] try to gain privilege by flipping bits in sensitive areas of memory. Since Synergy corrects all bit errors within a single chip, it can not only detect these attempts, but also correct them and be resilient to such attacks as long as they are localized to a single chip. In case such attacks cause bits in multiple chips to flip, then Synergy will detect it using the MAC, but be unable to correct and declare an attack.

V. Experimental Methodology

Performance simulations: For evaluating performance and memory power, we use USIMM [27], [28], a memory system simulator. We compare the performance of different configurations on the basis of IPC (Instructions per Cycle) normalized to the baseline configuration. Additionally, we evaluate the overheads in each configuration by quantifying the bloat in memory traffic due to secure execution, i.e. the additional memory accesses made by the application for security metadata and the bloat in memory traffic incurred for reliability, i.e. memory accesses for parity updates.

Reliability simulations: For evaluating reliability, we use FaultSim [29] a memory reliability simulator. We use a fault model based on real-world field studies from Sridharan et.al. [8] shown in Table I. We evaluate the reliability of the system by calculating the probability of system failure, i.e. the probability of the system encountering an uncorrectable error. We measure this by performing Monte-Carlo simulations on 1 billion devices over a 7 year lifetime and calculating the fraction of failed systems.

System configuration: We evaluate Synergy with a secure memory design using 56-bit counters for encryption, 64-bit MACs (AES-GCM based GMACs) and a Bonsai-style counter tree [6], [14] using 56-bit counters and 64-bit MACs (AES-GCM based GMACs). Our baseline secure memory design SGX_O using SECDED ECC-DIMM, is an optimized version of SGX that caches counters in both dedicated and last-level cache. We also compare against SGX that only caches counters in the dedicated cache. For consistency, we assume a 64-bit GMAC as the MAC design for SGX_O and SGX. We also separately compare against IVEC, that uses a MAC-Parity co-design with a non-Bonsai Merkle-Tree based integrity tree. Tables II and III provide more details regarding the configurations we evaluate.

Table III. Baseline system configuration

Number of cores	4
Processor clock speed	3.2GHz
Processor ROB size	192
Processor fetch, retire width	4
Last Level Cache (Shared)	8MB, 8-Way, 64B lines
Metadata Cache (Shared)	128KB, 8-Way, 64B lines
Memory bus speed	800MHz
DDR3 Memory channels	2
Ranks per channel	2
Banks per rank	8
Rows per bank	64K
Columns (cache lines) per row	128

Workloads: We evaluate Synergy with workloads from SPEC2006 [30] and GAP [26] benchmark suites. As Synergy is focused on reducing the application memory traffic, we present evaluations only on memory-intensive workloads from SPEC2006 (>1 memory access per 1000 instructions). From GAP, we use 6 representative workloads (Page Rank, Connected Components, Betweenness Centrality kernels with Twitter and Web data-sets). For each benchmark, we pick a representative slice of 1 billion instructions using Pinpoints. Our evaluations run the benchmarks in rate mode, i.e. each of the four cores runs the same copy of the benchmark. Additionally, we evaluate 6 mixed workloads generated by choosing random combinations of 4 benchmarks.

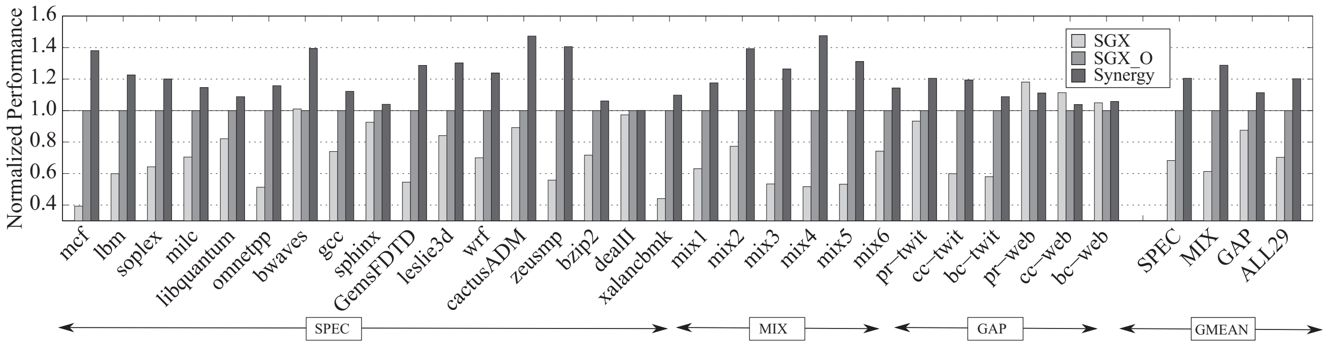


Figure 8: Performance (IPC) of SGX, SGX_O and Synergy normalized to SGX_O Baseline. Synergy improves the performance of secure execution by 20% compared to SGX_O. SGX has 30% lesser performance than SGX_O.

VI. Results

A. Impact on Performance

Figure 8 compares the performance of Synergy (using co-design) with SGX and SGX_O (without the co-design) all normalized to SGX_O's IPC (Instructions Per Cycle), for 29 workloads from across SPECint, SPECfp, GAP suites. Synergy improves the performance of secure execution by 20% compared to SGX_O baseline, while SGX has a performance slowdown of 30% compared to SGX_O.

Secure memories consume considerable memory bandwidth due to additional accesses for metadata (i.e. counters, integrity tree entries and MACs). Therefore, secure memory systems are more bandwidth-constrained than non-secure systems. As Synergy places the MAC in the ECC-chip and accesses it alongside data, it eliminates the bandwidth overhead of accessing the MAC. As a result, Synergy reduces the overall memory traffic and the corresponding queuing delay for the memory accesses. This enables Synergy to improve performance compared to SGX_O across all workloads.

SGX shows a considerable slowdown primarily because it uses a dedicated cache for the counters that are required on each data access for decryption and integrity verification. The small dedicated cache is unable to sustain the working set of counters, leading to considerable memory accesses for fetching counters and causing a considerable performance slowdown. Our baseline SGX_O caches counters in the last-level-cache, in addition to the dedicated cache. This reduces the memory traffic for counters and improves performance for SGX_O compared to SGX.

Graph workloads with the web dataset (pr-web, cc-web, bc-web) are an exception showing a slowdown for SGX_O compared to SGX. For these workloads, counters aggressively compete with data for last-level cache space, increasing data evictions and overall memory traffic in SGX_O. However, Synergy still shows speedup compared to SGX_O, as it doesn't require memory accesses for the MAC.

Synergy does not show any difference in performance for non-memory intensive workloads from SPEC2006 as they are bandwidth insensitive. As the memory traffic is negligible in the baseline (<1 memory access per 1000 instructions), a reduction in memory traffic due to Synergy does not have any perceptible impact on performance.

B. Analyzing Memory Access Bloat

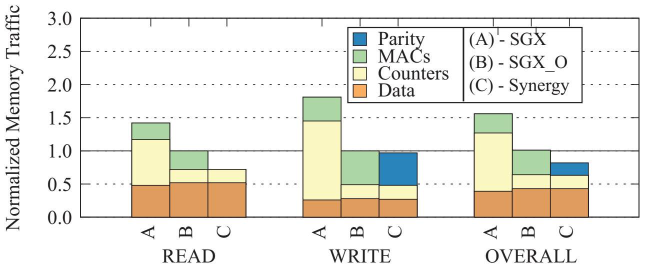


Figure 9: Memory traffic by type of access, normalized to SGX_O. Synergy reduces the accesses for MACs on reads and writes, but incurs accesses for Parity updates on writes.

Figure 9 shows the memory traffic (number of memory accesses per thousand instructions) for Synergy compared against SGX and SGX_O, normalized to baseline SGX_O. The memory traffic is split into four categories: (1) accesses to program-related data, metadata for security like (2) counters and (3) MACs, and metadata for reliability like (4) parity. We present this analysis for data reads, data writes and overall, showing benefits of Synergy for reads.

The memory traffic in SGX and SGX_O consists of accesses to program data, in addition to accesses for counters and MACs required for ensuring security (security bloat). There are no additional accesses towards reliability as the 9 chips in ECC-DIMM (8 data and 1 ECC chip) are accessed concurrently, providing ECC in the same access as Data. SGX on average incurs additional accesses for counters compared to SGX_O. This is because SGX caches counters in a small dedicated cache that is unable to store the entire working set of counters, leading to increased memory accesses. This effect is more pronounced for writes, as that requires updates to multiple levels of counters (encryption and integrity tree). SGX_O allows caching counters in last-level cache, reducing the accesses for counters, while retaining the accesses for the MAC. As a result, MACs become the dominant fraction of the security bloat in SGX_O.

Synergy does not require separate accesses for the MAC on Data reads and writes, due to its co-location of MAC and data. This reduces the security-bloat compared to SGX_O on both reads and writes. On writes, Synergy incurs an equivalent bloat for updates to parity, which is stored in a separate region of the memory. This leads to similar memory traffic compared to SGX_O on writes. Overall, Synergy reduces the memory accesses by 18%.

To avoid separate accesses for parity updates on data writes, co-location of both MAC and parity with Data is possible with custom DIMMs that provide 16 bytes of additional metadata per 64-byte cacheline. Such organizations may be used for future standards on reliable and secure memories.

C. Impact on System Power and Energy

Synergy impacts system energy due to reduction of extra memory accesses and change in the execution time compared to SGX and SGX_O. Figure 10 shows the power, performance, energy, and Energy-Delay product (EDP) for the different configurations, all normalized to SGX_O.

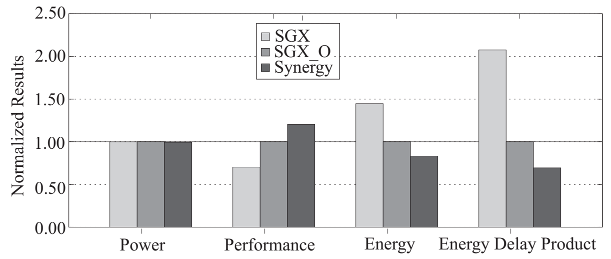


Figure 10: Power, Performance, Energy and System-EDP of SGX, SGX_O and Synergy.

As power is dependent on both energy consumption and execution time, it remains similar across the three configurations. SGX needs extra accesses for counters, consuming more energy per memory request (resulting in higher total energy). However, it also has higher execution time (reduced performance), leading to similar power as baseline SGX_O. On the other hand, Synergy reduces execution time and has less energy per access (due to avoided MAC accesses). Overall, Synergy reduces EDP by 31% compared to SGX_O.

D. Impact on Reliability

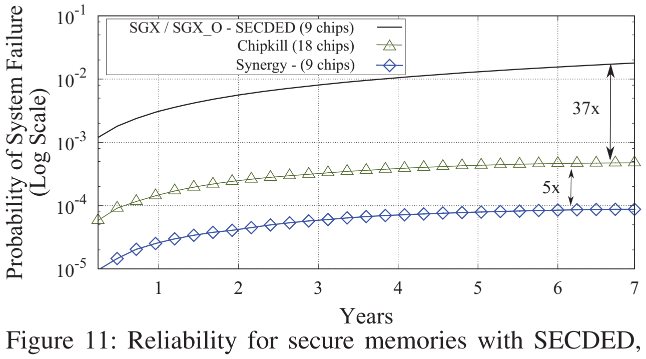


Figure 11: Reliability for secure memories with SECDED, Chipkill and Synergy. Synergy reduces probability of system failure by 185x compared to SECDED.

We compare reliability provided by Synergy against the SECDED protection available to SGX_O or SGX using ECC-DIMM (9 chips) and Chipkill (implemented with 18 chips over two channels). Figure 11 shows the probability of a system failure due to an error over a period of 7 years. SGX_O has a low level of reliability since SECDED can tolerate only single-bit failures. As compared to that, conventional Chipkill provides 37x reduction in failure probability, since it can tolerate 1 chip failure out of 18 chips. Synergy provides 5x reduction in probability of system failure compared to Chipkill (and 185x compared to SECDED), as it can correct 1 chip failure within 9 chips. This is in line with the observation that the probability of two chips being erroneous varies as the square of the number of chips that could potentially be faulty.

E. Sensitivity to Memory Organization

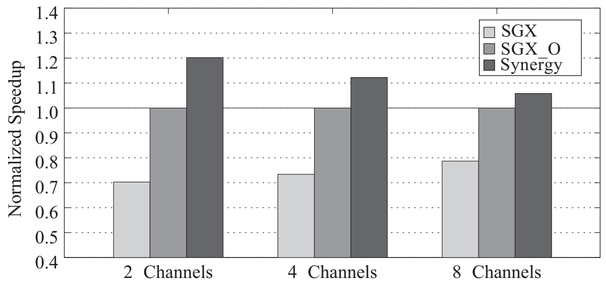


Figure 12: Performance benefits of Synergy as the number of channels is varied from 2 to 8

Figure 12 shows the performance of SGX, SGX_O and Synergy all normalized to SGX_O as the number of channels is increased from 2 (default) to 8. With increasing channels, the system becomes less bandwidth bound. This leads to higher tolerance of the security bloat, reducing the slowdown for SGX from 29% to 21%. Additionally, the MAC accesses of secure memory have a lesser impact on performance in the baseline, causing the performance improvement of Synergy to reduce from 20% to 6%.

F. Sensitivity to Counter Organization

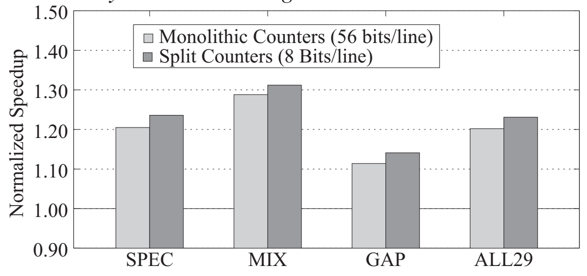


Figure 13: Speedup with Synergy vs SGX_O, using Monolithic Counters (default) and Split Counters.

We evaluate the benefits of using Synergy with an alternate secure memory design that uses Split Counters [17]. Split Counter design proposes using smaller per-line counters (8 bits per line) to improve performance, as opposed to monolithic 56-bit counter per line in our secure memory design. Figure 13 shows the speedup of Synergy with both monolithic counters (default) and Split Counters, compared against SGX_O using the same counter design. Synergy with Split Counters provides 3% additional speedup than monolithic counters, as this optimization further improves the cache-ability of the counters, making MACs a larger fraction of the security bloat in the baseline. Nonetheless, Synergy is effective for both counter organizations.

G. Sensitivity to caching Counters in LLC

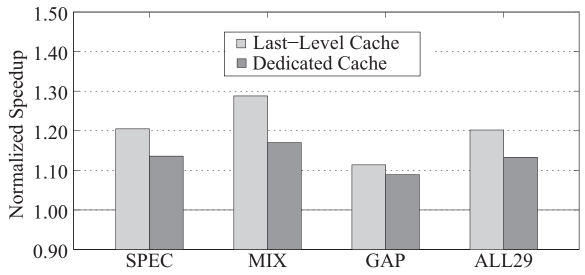


Figure 14: Synergy speedup when counters use both dedicated and last-level cache (default) vs only dedicated cache.

We evaluate the benefits of using Synergy with secure memory designs that either use both dedicated and last-level cache (LLC) for caching counters (like SGX_O), or use only dedicated cache for counters (like SGX). Figure 14 shows the speedup of Synergy in both configurations, normalized to SGX_O and SGX respectively. Synergy using only dedicated counter cache shows lesser performance benefits (13%) compared to Synergy using LLC for caching counters (20%). This is because counter accesses form a larger fraction of the memory traffic in the dedicated cache configuration, leading to MACs accesses forming a smaller portion of the memory traffic. As Synergy's performance benefit stems from reduction in MAC memory accesses, the configuration using only dedicated counter cache, shows a smaller speedup. Nevertheless, Synergy benefits both configurations.

VII. Related Work

We quantitatively compare against prior work IVEC, a proposal combining memory security and reliability that is closest to our proposal. We also qualitatively compare proposals for reliability-security co-design in other areas.

A. Co-design of Reliability and Security

1) Memory reliability-security co-design (IVEC): Prior work IVEC [10] has proposed co-designing reliability and security to provide Chipkill reliability for x4 Commodity DIMMs without ECC-chips. It proposes using MACs as error detection codes and using 4 bit - 4 byte parities per cacheline to correct 1 chip failure out of 16 chips (50x reliability compared to SECDED).

While IVEC is an excellent design providing strong reliability and security, it does not optimize for performance, which is a key impediment to adoption of secure memories. Furthermore, with reliability being an absolute requirement, ECC-DIMMs have become the de facto standard in datacenters [31]. For reasons of economies of scale, it is simpler even for secure datacenters to use ECC-DIMMs [32]. In this context, while IVEC proposes eliminating ECC-chips, Synergy re-purposes the ECC-chips for storing MACs and improves performance and Energy-Delay Product. Thus Synergy makes a stronger case for reliability-security co-design in memory systems.

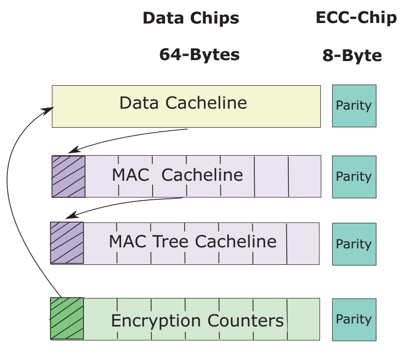


Figure 15: IVEC with x8 ECC-DIMM places parity in ECC-chip, not using ECC-chip bandwidth optimally.

Using IVEC with an ECC-DIMM, forces the placement of the parity in the ECC-chip. This results in sub-optimal usage of the ECC-chip bandwidth as the parity is not used in the common case (error-free) access, but only used for correction in the rare case an error is detected by the MAC. IVEC does not benefit from storing MACs in the ECC-chip as it uses a MAC tree design as shown in Figure 15. As the tree node is constructed by hashing multiple sibling MACs together, storing them in ECC chips across separate cachelines causes a forced over-fetch of subsequent cachelines during the tree traversal. Storing the counter associated with the data cacheline in the ECC-chip does not provide significant performance benefits either. This is because they are cached on-chip for pre-computation of the decryption pad with a high cache hit-rate [17]. Synergy is able to place the MACs in the ECC-chip to speedup secure execution, since it uses a Counter Tree based integrity tree where the MACs of Data cachelines are not a part of the tree.

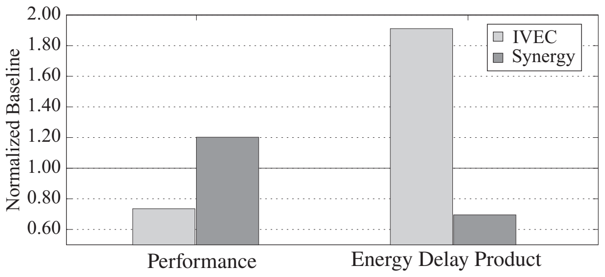


Figure 16: Performance and EDP of IVEC and Synergy, normalized to SGX_O.

As shown in Figure 16, IVEC suffers from a performance slowdown of 26% compared to SGX_O. This is because IVEC requires a sub-optimal non-Bonsai Merkle-tree as it assumes every entity in memory has a MAC, by design.[4]. A non-Bonsai tree design suffers from a slowdown due to higher cache contention for counters [14]. Furthermore, as IVEC needs to cache MACs in the LLC, it uses only a

[4]For replay attack protection, it is sufficient for the integrity tree to protect only counters out of (Data, Counter, MAC) [14]

dedicated cache for the Counters. As Synergy fetches MACs using ECC-bandwidth, it does not need to cache MACs allowing counters to be cached in LLC – thus improving performance compared to SGX_O by 20% and IVEC by 63%. On similar lines, Synergy reduces system EDP by 31% while IVEC increases it by 90% compared to SGX_O.

2) Other areas of research: Stinson [33] first studied Carter-Wegman MACs [21] in the context of error correction. Building on that, Dubrova et. al. [34] proposed a Carter-Wegman MAC for 5G Communication, capable of correcting 1-bit errors and avoiding retransmission power overheads. However, such SECDED code is insufficient for high reliability in memory systems. In contrast, our proposal Synergy provides Chipkill-level reliability. Other works [35], [36], [37], [38], [39] propose Approximate MACs for authentication of error-tolerant data like images, videos etc. However, they are restricted to enabling error-tolerant authentication and not capable of correcting errors.

B. Prior Work on Secure Memories

Providing memory integrity with keyed hashes and integrity trees is a well-researched area. Prior proposals [14], [17], [18], [19], [40], [41], [42] have optimized the integrity verification process with alternate tree designs or caching mechanisms. Synergy may be applied with these proposals by placing the Nonce (e.g. TEC-Tree [41]) or MAC (e.g. SGX Counter-Tree [6]) for the Data in the ECC-Chip, as long as the construction of the Tree does not require them to be stored in contiguous memory.

High-end Intel processors provide a feature called Software Guard Extensions (SGX) [6], [7] that provides secure memory enclaves for applications. While Synergy can improve the performance of secure execution in this context, it can also provide strong reliability for the region covered by security (enclave or entire memory).

PoisonIvy [15] and Authenticate-then-Write [43] propose mechanisms for speculative execution of unauthenticated data, to avoid the latency penalty of authentication on the critical path. However, these solutions still require memory accesses for performing integrity verification off the critical path. These designs would benefit from the bandwidth savings provided by Synergy. As these proposals do not possess a mechanism to recover architectural state on mis-speculations (i.e. errors), using Synergy with these proposals would prevent data-loss but still halt execution. If support for roll-back of architectural state on mis-speculation is added, Synergy can be completely integrated with these proposals.

Oblivious RAMs [44], [45], [46] or ORAMs address the side-channel of unencrypted memory access pattern. More recently, ObfusMem [47] and InvisiMem [48] have proposed smart-memory solutions to enable low-overhead ORAM guarantees. As long as these proposals use integrity protection [49], [50], Synergy's philosophy of symbiotic co-design of reliability and security may be applicable.

C. Prior Work on Memory Reliability

Several recent proposals [51], [52], [53], [54], [55], [56], [57], [58] have looked at providing memory reliability at low-cost. We discuss two proposals that are most closely related: Memguard [59] and LOT-ECC [12]. Memguard uses non-ECC DIMMs and provides strong reliability by storing hashes of data and check-pointing execution. Akin to MACs in Synergy, Memguard uses these stored hashes to detect errors. However, unlike Synergy which can simply use parity to fix failures, Memguard incurs check-pointing overheads to tolerate failures. LOT-ECC provides Chipkill using x8 DRAM-chips by using local error detection per line and a chip-level parity-based correction, similar to Synergy. However, LOT-ECC, as proposed, does not leverage any of the apparatus of secure memory to reduce area, power and performance overheads. By reusing MAC as a detection code, Synergy provides a much strong error detection than LOT-ECC without incurring any additional cost for local-error detection. Additionally, by avoiding additional accesses for MACs, Synergy improves performance of secure memory systems. To highlight the performance benefits of co-designing reliability and security, we compare LOT-ECC implemented on a secure memory against Synergy. Figure 17 compares the performance and EDP of LOT-ECC (with and without write coalescing) against Synergy. LOT-ECC incurs 15%-20% slowdown, whereas Synergy provides 20% speedup. Thus, Synergy has lower storage overhead, better energy-efficiency, and higher performance than LOT-ECC.

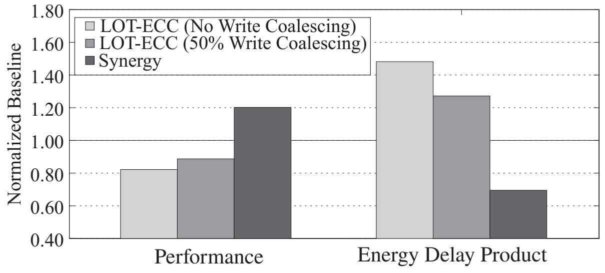


Figure 17: Performance and EDP of Secure Memory with LOT-ECC and Synergy, normalized to SGX_O.

VIII. Summary

Memory security and reliability are key requirements for building trusted data-centers. While existing commercial solutions design these two components independently, we propose a co-design to leverage the *Synergy* between them. This paper, called Synergy, has the following contributions:

1) A secure memory organization for a 9-chip ECC-DIMM storing MACs in the ECC-chip, that improves performance of secure execution by 20%.
2) A reliability-security co-design providing Chipkill-level reliability, that reduces the probability of system failure by 185x compared to SECDED.

We believe such designs can enable practical secure memories and facilitate their widespread commercial adoption.

Acknowledgment

We thank the anonymous reviewers for their comments and constructive inputs. We are also thankful to José Joao and Roberto Avanzi from ARM and members of our research lab for their invaluable feedback. This work was supported in part by NSF Grant 1526798 and a gift from Intel.

References

[1] J.A. Halderman *et al.*, "Lest we remember: cold-boot attacks on encryption keys," *CACM*, 2009.
[2] S.F. Yitbarek *et al.*, "Cold boot attacks are still hot: Security analysis of memory scramblers in modern processors," in *HPCA*, 2017.
[3] Y. Kim *et al.*, "Flipping bits in memory without accessing them: An experimental study of dram disturbance errors," in *ISCA*, 2017.
[4] M. Seaborn and T. Dullien, "Exploiting the dram rowhammer bug to gain kernel privileges," *Black Hat*, 2015.
[5] M. Becher *et al.*, "Firewire: all your memory are belong to us," *CanSecWest*, 2005.
[6] S. Gueron, "A memory encryption engine suitable for general purpose processors," *IACR Cryptology ePrint Archive*, 2016.
[7] V. Costan and S. Devadas, "Intel sgx explained." *IACR Cryptology ePrint Archive*, 2016.
[8] V. Sridharan and D. Liberty, "A study of dram failures in the field," in *SC*, 2012.
[9] B. Schroeder *et al.*, "Dram errors in the wild: a large-scale field study," in *SIGMETRICS*, 2009.
[10] R. Huang and G.E. Suh, "Ivec: Off-chip memory integrity protection for both security and reliability," in *ISCA*, 2010.
[11] T.J. Dell, "A white paper on the benefits of chipkill-correct ecc for pc server main memory," *IBM Microelectronics Division 11*, 1997.
[12] A.N. Udipi *et al.*, "Lot-ecc: Localized and tiered reliability mechanisms for commodity memory systems," in *ISCA*, 2012.
[13] P.J. Nair *et al.*, "Xed: exposing on-die error detection information for strong memory reliability," in *ISCA*, 2016.
[14] B. Rogers *et al.*, "Using address independent seed encryption and bonsai merkle trees to make secure processors os- and performance-friendly," in *MICRO*, 2007.
[15] T.S. Lehman *et al.*, "Poisonivy: Safe speculation for secure memory," in *MICRO*, 2016.
[16] H. Lipmaa *et al.*, "Comments to nist concerning aes modes of operation: Ctr-mode encryption," 2000.
[17] C. Yan *et al.*, "Improving cost, performance, and security of memory encryption and authentication," in *ISCA*, 2006.
[18] B. Gassend *et al.*, "Caches and hash trees for efficient memory integrity verification," in *HPCA*, 2003.
[19] G.E. Suh *et al.*, "Efficient memory integrity verification and encryption for secure processors," in *MICRO*, 2003.
[20] R.C. Merkle, "Protocols for public key cryptosystems," in *S&P (Oakland)*, 1980.
[21] M.N. Wegman and J.L. Carter, "New classes and applications of hash functions," in *FOCS*, 1979.
[22] B. Schroeder and G. Gibson, "A large-scale study of failures in high-performance computing systems," *IEEE Trans Dependable Secure Comput.*, 2010.
[23] C. Chen and M. Hsiao, "Error-correcting codes for semiconductor memory applications: a state-of-the-art review," *IBM JRD*, 1984.
[24] R. Bose and D. Ray-Chaudhuri, "On a class of error correcting binary group codes," *Information and Control*, 1960.
[25] I.S. Reed and G. Solomon, "Polynomial codes over certain finite fields," *SIAM J Appl Math*, 1960.
[26] S. Beamer *et al.*, "The gap benchmark suite," *arXiv*, 2015.
[27] N. Chatterjee *et al.*, "Usimm: the utah simulated memory module," *University of Utah, Tech. Rep*, 2012.
[28] (2012) Memory scheduling championship (msc).
[29] P.J. Nair *et al.*, "Faultsim: A fast, configurable memory-reliability simulator for conventional and 3d-stacked systems," in *ACM-TACO*, 2015.
[30] "Spec cpu2006 benchmark suite," in *Standard Performance Evaluation Corporation*.
[31] J. Hamilton, "You really do need ecc memory - perspectives," http://perspectives.mvdirona.com/2009/10/you-really-do-need-ecc-memory/.
[32] "Intel data center block for secure enclaves," https://www.intel.com/content/www/us/en/data-center-blocks/business/secure-enclaves-blocks.html.
[33] D.R. Stinson, "On the connections between universal hashing, combinatorial designs and error-correcting codes," *Congressus Numerantium*, 1996.
[34] E. Dubrova *et al.*, "Error-correcting message authentication for 5g," in *MOBIMEDIA*, 2016.
[35] R. Ge *et al.*, "Approximate message authentication codes for n-ary alphabets," *IEEE Trans. Inf. Forensic Secur.*, 2006.
[36] L. Xie *et al.*, "Approximate image message authentication codes," *IEEE Trans. Multimed*, 2001.
[37] S. Xiao and C.G. Boncelet, "Efficient noise-tolerant message authentication codes using direct sequence spread spectrum technique," in *CISS*, 2006.
[38] O. Ur-Rehman *et al.*, "Error correcting and weighted noise tolerant message authentication codes," in *ICSPCS*, 2011.
[39] D. Tonien *et al.*, "Unconditionally secure approximate message authentication," in *IWCC*, 2009.
[40] J. Lee *et al.*, "Reducing the memory bandwidth overheads of hardware security support for multi-core processors," *IEEE Trans. Comput*, 2016.
[41] R. Elbaz *et al.*, "Tec-tree: A low-cost, parallelizable tree for efficient defense against memory replay attacks," in *CHES*, 2007.
[42] W.E. Hall and C.S. Jutla, "Parallelizable authentication trees," in *SAC*, 2005.
[43] W. Shi and H.H.S. Lee, "Authentication control point and its implications for secure processor design," in *MICRO*, 2006.
[44] O. Goldreich and R. Ostrovsky, "Software protection and simulation on oblivious rams," *JACM*, 1996.
[45] E. Stefanov *et al.*, "Path oram: An extremely simple oblivious ram protocol," in *CCS*, 2013.
[46] M. Maas *et al.*, "Phantom: Practical oblivious computation in a secure processor," in *CCS*, 2013.
[47] A. Awad *et al.*, "Obfusmem: A low-overhead access obfuscation for trusted memories," in *ISCA*, 2017.
[48] S. Aga and S. Narayanasamy, "Invisimem: Smart memory defenses for memory bus side channel," in *ISCA*, 2017.
[49] L. Ren *et al.*, "Integrity verification for path oblivious-ram," in *HPEC*, 2013.
[50] C.W. Fletcher *et al.*, "Freecursive oram: [nearly] free recursion and integrity verification for position-based oblivious ram," in *ASPLOS*, 2015.
[51] D.J. Palframan *et al.*, "Cop: To compress and protect main memory," in *ISCA 2015*.
[52] J. Kim *et al.*, "Frugal ecc: Efficient and versatile memory error protection through fine-grained compression," in *SC*, 2015.
[53] D.H. Yoon and M. Erez, "Virtualized and flexible ecc for main memory," in *ASPLOS*, 2010.
[54] J. Kim *et al.*, "Bamboo ecc: Strong, safe, and flexible codes for reliable computer memory," in *HPCA*, 2015.
[55] X. Jian and R. Kumar, "Adaptive reliability chipkill correct (arcc)," in *HPCA*, 2013.
[56] X. Jian *et al.*, "Low-power, low-storage-overhead chipkill correct via multi-line error correction," in *SC*, 2013.
[57] P.J. Nair *et al.*, "Citadel: Efficiently protecting stacked memory from large granularity failures," in *MICRO*, 2014.
[58] X. Jian *et al.*, "Parity helix: Efficient protection for single-dimensional faults in multi-dimensional memory systems," in *HPCA*, 2016.
[59] L. Chen and Z. Zhang, "Memguard: A low cost and energy efficient design to support and enhance memory system reliability," in *ISCA*, 2014.