1 INTRODUCTION AND MOTIVATION

Main-memories are prone to attacks [1], [4], [12] that allow an adversary to take control of the system by reading and tampering memory-contents. Commercial solutions like Intel’s Software Guard Extensions (SGX) [3] and AMD’s Secure Memory Encryption (SME) [5] attempt to secure memory against attacks. However, providing security requires accessing metadata, resulting in storage and performance overheads. In navigating this performance-security trade-off, SGX and SME end up at different ends of the spectrum as shown in Fig 1.

![Fig. 1. Performance vs Security Trade-off in Secure-Memory Designs.](image-url)

To avoid performance overheads, AMD-SME only provides encryption for the entire memory, which has been shown broken under fault-injection attacks [2]. On the other hand, Intel-SGX provides encryption, integrity-check, and replay-attack protection and is robust against such attacks, but incurs considerable performance overhead [6], [9] and hence only provides 96MB of secure memory. In this work, we explore a compromise that provides a probabilistic guarantee of memory-integrity for the entire memory with minimal performance impact.

All existing works detect a replay-attack within single-cacheline replay. However, for majority of data in memory, it may be sufficient to detect replay-attempts probabilistically, especially when an attack requires multiple attempts before being successful (e.g. fault injection attack on encrypted memory [2] has a probability of success between 10-50%). Such an attack can be thwarted if any one of the attempts is detected. Therefore, in this work, we explore the performance-benefits achievable with probabilistic detection of replay-attacks. We describe a Probabilistic Integrity Tree (ProbTree) that detects a replay-attack with 99% probability within replay of 7 cachelines, and show it improves performance by 14% while requiring 16x less storage compared to state-of-the-art VAULT [10].

2 BACKGROUND: COMPACT INTEGRITY-TREES

State-of-the-art integrity-trees are constructed over the encryption counter footprint, to prevent the replay of encryption counters and subsequently replay of the {data, counter} tuple. The tree consists of many levels, with a node at each level containing n-entries – each entry in a node prevents the replay of a lower-level node. Prior works have proposed packing more entries per node (higher tree-arity or fan-in per node) to achieve a shorter tree, with fewer levels that do not fit in the on-chip cache. VAULT [10] is a tree with variable-arity, 32-ary at the base-level and 16-ary above, whereas MorphTree [8] is 128-ary tree. More details about these trees may be obtained from the respective papers. In the next section, we demonstrate a 512-ary integrity-tree.

3 KEY IDEA: PROBABILITY INTEGRITY-TREE

The insight behind ProbTree is an integrity-tree with 1-bit hashes (e.g. truncated-SHA) constructed over the base of encryption-counters (bonsai-style [7]), that allows 512-signatures per tree-node as shown in Fig 2. This results in a 2-level tree – level-1 signatures protecting the encryption counters have a footprint of 512KB and are stored in-memory. Whereas level-2 signatures protecting level-1 nodes have a footprint of 1KB and are stored on-chip, safe from replay. Level-1 is stored after encryption.

![Fig. 2. For a 128GB memory, (a) VAULT has 16-32 arity, resulting in tree size of 68MB. (b) ProbTree has 512-arity (with 512 x 1-bit hashes/tree-node) resulting in 4MB tree-size.](image-url)

Security Analysis: We assume the attacker is only capable of taking periodic snapshots of memory and replaying arbitrary cachelines from a past snapshot. If an attacker replays a single counter-cacheline without changing level-1 node (similar analysis follows for level-1 replay), the probability of level-1 signature-mismatch on a subsequent read is 50% (p). Subsequent n-replay-attempts causes the cumulative probability of attack-success to drop to $p^n$. Thus,
the probability of detection \((1 - p^n)\) surpasses 99% within 7 replay attempts. Between two snapshots, there may be counter-cachelines that changed but their level-1 nodes are unchanged. However, the probability of such an event for a level-1 node is \((1 - p^x)\), where \(x\) is sum of writes received to its child-counter cachelines. This is small-enough if the attacker is limited to taking snapshots sufficiently spaced out in time.

**Performance Analysis:** We evaluate ProbTree using split counters [11] for encryption (64-counters/cacheline), 64-bit MAC-like SYNERGY [9], 128KB dedicated metadata-cache (for 4 cores) [6], [10], 8MB Shared-LLC and 128GB DRAM. We compare performance of our ProbTree, a 512-ary integrity-tree against a baseline using 16-32-ary VAULT (as shown in Fig.2), a 8-ary SGX-like Tree design, a 128-ary MorphTree [8] and an ideal NoTree design. We assume optimistic models for VAULT and MorphTree without any counter-overflows. In reality, these designs would have lower performance due to overheads of memory accesses for servicing the overflows. We evaluate our designs with 28 memory intensive workloads from SPEC2006 and GAP.

![Fig. 3. (a) Performance – ProbTree has 14% speedup vs VAULT and is within 8% of ideal-NoTree. (b) Metadata accesses per data access – ProbTree halves it compared to VAULT.](image)

As shown in Fig.3(a), ProbTree has 14% speedup compared to VAULT and bridges two-thirds of the gap between VAULT and ideal-NoTree. This is because the integrity-tree traversal in ProbTree accesses one level of the integrity-tree from memory as shown in Fig.3(b), as compared to VAULT that accesses 3 levels. SGXTree suffers a 10% slowdown compared to VAULT as it accesses up to 5 levels. In comparison, MorphTree would access 2 levels of the tree and hence provide 9% speedup. It is important to note that the performance difference between MorphTree and ProbTree becomes imperceptible at smaller memory sizes (e.g. 16 GB), where both designs have only 1-level of the tree that does not fit in cache and requires memory accesses.

**4 Conclusion**

Current secure memory designs are at two extremes: they provide either no replay-attack protection (AMD-SME) or strong replay-attack protection but with significant performance overheads (Intel-SGX). In comparison, our Probabilistic Integrity-Tree provides strong detection of replay attacks (within an attack to few lines) while significantly reducing performance overheads. This is an initial study and we continue to explore the performance-security trade-off.

**References**


