“Instant On” Encrypted Non-Volatile Main Memories

Samuel Thomas

March 7, 2021

Abstract

The intersection of encrypted memory and non-volatile main memories is an emerging area of architecture research. This is largely because non-volatile main memories are subject to an attack that their DRAM counterparts are not; namely, an attacker can cut power to the system and tamper with the contents of memory. As such, encrypted memory must employ counter-mode encryption and Bonsai Merkle Trees to check the integrity of the memory upon powering on the system. The trees are large, however, so the task is incredibly expensive. State-of-the-art work either takes several hours to reconstruct this tree [8], which is an unreasonable amount of time for reboot, or causes a 2% normal execution overhead, which can be incredibly costly for candidate applications of this architecture (i.e., cloud computing services).

This work proposes a new scheme for fast recovery of integrity-back encrypted non-volatile main memory systems. The solution should provide “instant” power-on without causing any normal execution overhead by introducing a novel *partial recovery scheme*. This scheme should provide at least 8x speedup and requires no modifications to normal execution.

1 Background

In the architecture community, encrypted memory and non-volatile main memories have emerged as two topics of interest. However, their combination introduce non-trivial security challenges. In particular, consider the following attack: an attacker turns the power of the system off and modifies bits within the non-volatile main memory unit. Seeing as this memory retains its state without power in non-volatile main memory, sensitive system code and application data may have been tampered. As a result of this attack, the contents of main memory cannot necessarily be trusted when they are loaded on-chip, motivating why encrypted memory alone for non-volatile memories [4] is not sufficient.

As a result, state-of-the-art work pertaining to encrypted non-volatile main memories relies on *integrity protected* encrypted memory. Namely, metadata is stored on-chip in such a way that any piece of data loaded from the main memory unit can be checked to see that it hasn’t been unexpectedly tampered. To do so, counter-mode encryption is coupled with a Bonsai Merkle Tree (BMT) [6, 7].

As the name suggests, counter-mode encryption is a per-page encryption scheme where each page is assigned a series of major and minor counters as well as other metadata. The metadata is combined with an on-chip private key to encrypt and decrypt data in constant time when they are loaded from memory. What’s important to note about the counters are that they are unique to each page and reflect the state of the contents of the page. That is, each time the contents of the page are updated, so too are the metadata counters.

The metadata counters are stored in memory, and they make up the leaves of the BMT. Each interior node of the tree is made up of a combination of its children’s hashes. As such, the root of the tree is a series of hashes, but its value is unique to the state of the overall memory unit. Thus, the root of the tree is stored and modified on-chip along with any metadata counter updates to ensure that any data loaded from memory hasn't been tampered with. Furthermore, the state of the entire memory unit can be verified by recomputing the tree and determining whether or not the computed root and the stored root are equivalent. If the memory is non-volatile, the root must also be non-volatile to check the integrity of the memory unit upon power-on.

Recovery of counter-mode encrypted non-volatile main memories has been explored in [8, 9, 1]. Anubis [9], Osiris [8], and Phoenix [1] are papers concerned with the notion that strict persistence [5] of metadata is highly inefficient for the average case and, as such, will not be common in most real non-volatile main memory systems. Approaching the problem in this way, however, does introduce a new issue. Namely, suppose a series of metadata updates occur on-chip before they are flushed to main memory, but the data has been flushed to main memory before the system crashes. In such a situation, the state of the metadata and data are in different states in main memory. As such, these papers propose recovery techniques related to predicting differences in metadata while still detecting corruption of main memory.

2 Motivation

Recomputing the BMT from scratch is an incredibly costly procedure. It is reasonable for a PCM-based non-volatile main memory to be 4TB due to its density compared to DRAM. The resulting tree is incredibly large and recomputing it in its entirety has been estimated to take on the order of "several hours" [9]. This is completely unreasonable for the candidate applications of these procedures, such as cloud computing servers, as it would cost them millions of dollars a year.

Previous work in [9] address this issue by maintaining "shadow tables" of metadata during execution, which allows for fast recovery time but adds a 2% overhead to normal execution. This average-case overhead is similarly unreasonable for cloud computing companies, which charge customers performing computationally expensive tasks on their servers by execution. For example, training a machine learning model that should take a week to train would require more than 5 hours additional hours of execution, which is significant enough dissuade a customer. As such, a different approach must be taken in order to achieve fast recovery speeds if the architecture is to be deployed in the real world.

3 Problem Statement

Although these systems provide working solutions to securely recovering non-volatile main memory units on power on, there either exists significant (about 2%) overhead in normal execution [9] or in recovery [8] (could take "several hours"). Such compromises are not reasonable in real-world applications of non-volatile encrypted memory architectures.

This proposal proposes a solution that enables intergity-backed encrypted non-volatile memory to have instant power-on while avoiding normal execution overhead.

4 Solution

In existing work, atomicity of updates to the BMT are implemented via redo-logging to ensure that the operation is an all-or-nothing transaction [8, 9, 1]. As such, previously proposed architectures assume a set of persistent registers on-chip (explicitly, in the memory controller). These registers store data to be written to memory, the appropriate counters for encryption, the nodes in the BMT that will be or have been effected, a temporary root (if the value is updated), and the permanent root. Refer to [9] for further details of the transaction protocols.

The proposed solution works from a key observation that one such register, namely the register tracking and updating BMT nodes that will be or have been effected, stores nodes other than the BMT root on-chip. Furthermore, seeing as these nodes are stored non-volatilely on-chip (within the memory controller), their values *can* be trusted on system reboot.

Given this observation, this proposal will work towards the novel idea of *partial recoverability of se-*

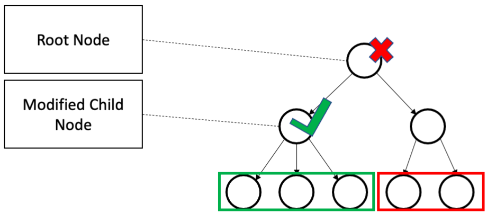


Figure 1: This figure describes the case in which the Bonsai Merkle Subtree root is verified but the root is not. This would imply that the green data could be recovered whereas the red data has been corrupted.

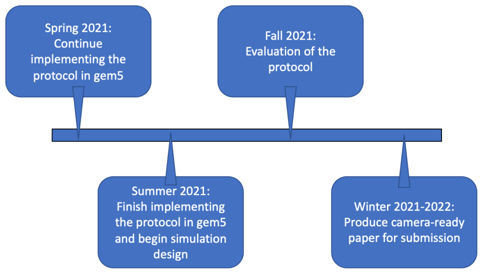
cure memory. That is, integrity checks are no longer require that an entire BMT be reconstructed for partial recoverability. This is done by modifying the integrity checking algorithm from the reconstruction of the entire BMT to reconstructing one stored subtree before the rest of the tree. That is, the non-root BMT node will be stored on-chip as the root, and reconstruct that tree. Then, reconstruct the rest of the tree.

The benefit to this approach is two-fold. To see this, consider the two possible outcomes from the subtree verification process. Suppose that the validation of the subtree fails. In this case, we have detected that the memory unit has been corrupted without having to recompute the entire BMT. This will save at least 50% of computation time if the BMT is binary and the stored node is the immediate child of the root. However, these trees are often 8-ary and by storing BMT nodes closer to the data level, recomputation of the tree will be even faster because the tree is smaller.

Now, suppose the validation of the subtree succeeds. It must be the case that this subset of the data has *not* been corrupted. As such, the system may be able to reboot with only a subset of the memory available. Furthermore, if the reconstruction of the rest of the tree fails, part of the memory unit may be recoverable to the user rather than needing to lose all memory in the system.

So far, this discussion has been contained to a fast hardware solution to the problem. However, it is entirely possible that software libraries could be developed that cater to the needs of the user as well. In a real deployment of the system, there will be certain data that are of a higher sensitivity than others. As such, it is possible that a custom `malloc` or `mmap` function could be developed for this architecture that guarantees that the allocated address will be recovered first. It would be interesting to explore and analyze different applications of the architecture to discover the proportion of security-critical memory occupies the overall space. Naturally, this would impact the balance of how much memory should be lazily recovered first. Furthermore, these applications and observations may impact the hardware recovery scheme algorithm.

5 Approach


Spring 2021: Continue implementing the protocol in gem5
Fall 2021: Evaluation of the protocol
Summer 2021: Finish implementing the protocol in gem5 and begin simulation design
Winter 2021-2022: Produce camera-ready paper for submission

This protocol will be implemeneted in the Gem5 [2] architecture simulator. It will be important to perform an evaluation on recovery time under an arbitrary power failure. Support for this type of evaluation is provided in Gem5, using ARM architecture support. However, it will be important to demonstrate correctness in the work in addition to merely providing metrics for this type of recovery.

With regards to recovery, it will also be important to show *how much memory* has been recovered over time, so as to evaluate "amount of data to recover" versus "time to recover said data."

Furthermore, it will be important to ensure that this protocol does not add average-case overhead, so this work will be evaluating the protocol using the memory-intensive benchmarks from the SPEC2007 suite [3], which is similar to prior work [9, 8].

References

[1] Mazen Alwadi et al. "Phoenix: Towards Ultra-Low Overhead, Recoverable, and Persistently Secure NVM". In: *IEEE Transactions on Dependable and Secure Computing* (2020).

[2] Nathan Binkert et al. "The gem5 simulator". In: *ACM SIGARCH computer architecture news* 39.2 (2011), pp. 1–7.

[3] James Bucek, Klaus-Dieter Lange, and Jóakim v. Kistowski. "SPEC CPU2017: Next-generation compute benchmark". In: *Companion of the 2018 ACM/SPEC International Conference on Performance Engineering.* 2018, pp. 41–42.

[4] Siddhartha Chhabra and Yan Solihin. "i-NVMM: a secure non-volatile main memory system with incremental encryption". In: *2011 38th Annual international symposium on computer architecture (ISCA).* IEEE. 2011, pp. 177–188.

[5] Steven Pelley, Peter M Chen, and Thomas F Wenisch. "Memory persistency". In: *2014 ACM/IEEE 41st International Symposium on Computer Architecture (ISCA).* IEEE. 2014, pp. 265–276.

[6] Brian Rogers et al. "Using address independent seed encryption and bonsai merkle trees to make secure processors os-and performance-friendly". In: *40th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO 2007).* IEEE. 2007, pp. 183–196.

[7] Gururaj Saileshwar et al. "Morphable counters: Enabling compact integrity trees for low-overhead secure memories". In: *2018 51st Annual IEEE/ACM International Symposium on Microarchitecture (MICRO).* IEEE. 2018, pp. 416–427.

[8] Mao Ye, Clayton Hughes, and Amro Awad. *Osiris: A Low-Cost Mechanism to Enable Restoration of Secure Non-Volatile Memories.* Tech. rep. Sandia National Lab.(SNL-NM), Albuquerque, NM (United States), 2018.

[9] Kazi Abu Zubair and Amro Awad. "Anubis: ultra-low overhead and recovery time for secure non-volatile memories". In: *Proceedings of the 46th International Symposium on Computer Architecture.* 2019, pp. 157–168.