CIAMH : Confidentiality, Integrity and Authentication across the Memory Hierarchy

Karim Ait Lahssaine and Olivier Savry

Univ. Grenoble Alpes, CEA, Leti, F-38000 Grenoble, France *
Email: firstname.name@cea.fr

Abstract

In this paper, we propose CIAMH, a hardware countermeasure that ensures the confidentiality, integrity and authenticity of data from DRAM to CPU registers, via L1D and L1I cache memories. Data confidentiality is ensured through encryption in DRAM and masking in caches. Integrity is guaranteed by associating integrity tags with data and by checking these tags at each level of the hierarchy to detect data corruption. The authentication of data is facilitated by authenticated encryption in DRAM and by the presence of integrity tags that are dependent on a unique key. These mechanisms have been designed to mitigate attacks such as RowHammer, fault injection and side-channel attacks throughout the memory hierarchy. The CIAMH has been implemented in relation to the NaxRiscv core, thus enabling it to be modular for the user. The RTL generated can easily incorporate part or all of the countermeasure, depending on the specific use case.

Introduction

In recent decades, there has been a considerable enhancement in the performance of processors. Nevertheless, the memory hierarchy has come to the fore as a pivotal bottleneck within the architecture, thereby impeding progress. Ensuring the security of this hierarchy poses a significant challenge, as any countermeasure that introduces even a minimal latency would result in a substantial loss of system performance.

In order to minimise the overhead, while guaranteeing a chain of trust for the data, several technological choices have been made and will be presented in a first step. The changes made to the architecture of the memory hierarchy will also be presented. The subsequent presentation will then cover the implementation of the architecture with a 64-bit RISC-V core, its modularity, and the corner cases that must be managed.

CIAMH Architecture

In accordance with the memory hierarchy, the proposed countermeasure involves the encryption and authentication of the DRAM, the masking of data in the L1 cache, and the association of an integrity tag with it. The registers are then masked, and a register file is added for the masks.Beyond DRAM, the decision is taken to mask rather than encrypt, with the aim of minimising the impact on memory access latency.

To traverse between these memory levels, modules are incorporated to oversee encryption, masking, and the calculation of the various integrity tags (figure 1). These modules also verify the integrity and authenticity of memory transactions and initiate an alert in the event of corrupted data.

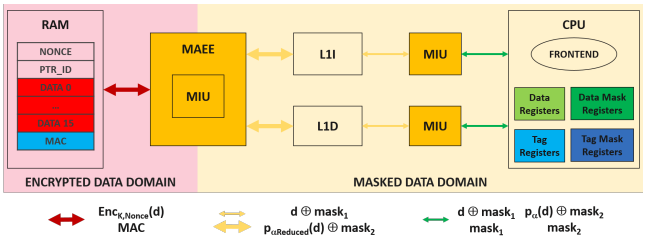

Figure 1: *CIAMH Architecture*

MAEE The first module is the Memory Authenticated Encryption Engine[1].

Write : MAEE unmasks and encrypts the cache lines, then associates a MAC (Message Authentication Code) with them. In order to limit the latency overhead, a lightweight authenticated encryption algorithm is employed, namely Subterranean 2.0[2]. The words on the cache line are unmasked and encrypted as they arrive, one by one, in the MAEE.

Read : The cache line originating from the DRAM is decrypted and subsequently masked. For each word, an integrity tag is calculated in parallel and also masked. Finally, at the conclusion of the process, the MAC is checked.

The transition between encrypted and masked data is made without revealing the data in plaintext, thus avoiding any breach of confidentiality.

MIU The Masking and Integrity Unit (MIU) in the MAEE is responsible for generating masking and integrity tags. The MIU generates two masks (data and tag masks) by utilising a secret key, the address of

*This work was funded thanks to the French national program "Programme Investissement d'Avenir IRT Nanoelec" ANR-10-AIRT-0005.

Figure 2: *Masked cache line (16 * 64-bit data) with integrity tags and metadata*

Figure 3: *Data masked in cache (64 bits) after modification of a part following a store*

the data, its ASID[1], a ptr_id[2], and an IV[3] as inputs. If only the IV changes, biased masks will be generated. However, Talaki et al.[3] have demonstrated that the masks generated leak low mutual information from side channel traces in the noisy environment of a processor.

In the context of integrity tags, a homomorphic permutation function is employed. This function involves the permutation of data d according to a key α, resulting in the generation of its tag $p_\alpha(d)$. It should be noted that the alpha key is unique to each user, thereby ensuring both authenticity and integrity.

Furthermore, a distinct key is utilised on both the CPU and the cache sides, with the objective of differentiating the integrity of these two components. The key is changed, and the integrity is verified in the MIUs at the interface.

In order to reduce memory overhead, masks are not stored in the cache, the integrity tag stored is compressed (32-bit MSB xor 32-bit LSB), and only the metadata needed by the MIU is cached (figure 2). MIU modules between caches (L1I and L1D) and the CPU are required in order to recalculate masks and tags.The masks change with each store as the IV is updated.

Finally, the CPU stores the masked data, its mask, the masked integrity tag and its mask in these registers.The masked data can then potentially be processed by a masked ALU.

Implementation

In order to evaluate the CIAMH countermeasure, a decision was taken to combine it with the RISC-V core, NaxRiscv. The rationale behind this choice was the modularity of its architecture, a feature facilitated by the SpinalHDL language.

However, the implementation process was accompanied by the emergence of numerous challenges, necessitating modifications to the countermeasure.

Initialisation At the initiation of the boot sequence, the DRAM must be initialised with encrypted data and their MACs; however, the data is located in the Flash memory.A firmware in the bootloader will then copy the data to the DRAM.The data is first transferred from the Flash memory to the CPU with uncached loads, and subsequently stored in the cache and then in the DRAM when cache lines are evicted, passing through the various modules (MIU, MAEE).

However, as we are using a write-back cache (write-allocate), to make a cached store, we first need to load the line. Nonetheless, upon loading the line, the MAEE will detect an incorrect MAC, as the DRAM remains uninitialised. Consequently, while the cache line is loaded, it is initialised to $0 \oplus m_1$. Thereafter, the line becomes available for stores.

Byte, Half and Word Store As the mask undergoes modification with each store, a problem arises when a 64-bit masked data item is not stored in its entirety. The CPU then transmits the part to be modified (byte, half or word) to the cache. This part is masked with a new mask. However, the remaining unmodified double word must have its mask updated with the new mask using transmasking (figure 3). A similar but more complex mechanism is also in place to manage the updating of integrity tags.

Modularity The modularity of the architecture enables users to select the protection of DRAM and/or caches according to their specific use cases. The SoC is generated automatically, with just the modules and modifications required, depending on two parameters: *withMIU* and *withEncryption*.

Performance CIAMH was synthesised using Yosys for Xilinx, associated with NaxRiscv and a L1 cache. The overhead in LUT is as follows: 1.4% for the MAEE alone, 63.5% for the MIU alone, and 66.7% with the two countermeasures. Furthermore, a flip-flop overhead of 0.4% is observed for the MAEE alone, 54.1% for the MIU alone, and 55% with the two countermeasures. The MIU countermeasure exerts the most significant influence on the SoC, by augmenting the logic and memory from the core to the caches.

Perspectives In the future, further obstacles must be surmounted, including the management of atomic operations, in order to ensure the functionality of a system running Linux. A security analysis on FPGAs and ASICs is also required to ascertain whether CIAMH can mitigate the aforementioned attacks, including side-channel attacks.

References

[1] Karim Ait Lahssaine and Olivier Savry. “Memory Authenticated Encryption Engine for a RISC-V processor”. In: *RISC-V Summit Europe* (June 2023).

[1] Address Space Identifier
[2] Optionally used for memory tagging purpose
[3] Initialisation Vector

[2] Joan Daemen et al. “The Subterranean 2.0 Cipher Suite”. In: *IACR Transactions on Symmetric Cryptology* 2020.S1 (June 2020), pp. 262–294.

[3] Ezinam Talaki. “A memory hierarchy protected against side-channel attacks”. PhD thesis. Université Grenoble Alpes, Sept. 2022, p. 87.