Efficient Memory Encryption for Neural Network Accelerators

George-Alexandru Stoian, Pavlos Aimoniotis, Per Ekemark, Xiaoyue Chen, Stefanos Kaxiras
Uppsala University, Sweden
george-alexandru.stoian.6273@student.uu.se, {pavlos.aimoniotis, per.ekemark, xiaoyue.chen, stefanos.kaxiras}@it.uu.se

***Abstract*—Modern IoT and edge devices, by their nature, are particularly vulnerable to hardware attacks. Providing data protection using "Secure Enclaves" and general lightweight cryptography techniques incurs an inordinately large performance, energy, and storage overhead cost. To accomplish efficient data protection, novel architecture designs are required. In this work, we focus on neural network accelerators running in inference phase, and we propose a design that reduces the latency and metadata cost incurred by memory encryption. The key concept of our work focuses on moving the security metadata from block granularity to tensor granularity, and using existing cryptography primitives cores in order to showcase that we can reduce both the storage required and hide the encryption latencies by leveraging the deterministic nature of the feedforward phase.**

I. Introduction

Secure computing requires devices being safe from both software and hardware attacks. A typical approach to hardware security is to guard sensitive data by keeping it in a "Secure Enclave" that encapsulates several techniques developed in the past two decades. Different vendors have similar implementations of the commercially viable subset of fundamental concepts constituting a "Secure Enclave" (Intel SGX [1], Vault [3], RISC-V Keystone [2]). In such enclaves the attacker should not be able to observe the data (confidentiality) or modify them without being detected (integrity). For confidentiality, counter-mode encryption is employed, where memory blocks are encrypted along with a monotonically increasing counter. Blocks are encrypted upon a write instruction, and decrypted upon a read instruction from the memory. For integrity, message authentication codes (MAC) are used along with the encryption. Intel SGX [1] uses 56-bit counters and 56-bit MAC for each memory block, introducing significant storage overhead.

II. Overview

In this work, we focus on machine learning accelerator embedded devices that deal with Convolutional Neural Networks (CNN) like models to infer a result (inference phase). The goal is to ensure that the model parameteres (i.e., weights) and the inference process are protected. Self-driving cars are an application where such security is crucial. The model weights are an expensive asset shipped with every car and the result of the inference process is integral to whether the car can be called self-driving or not.

By making use of the domain knowledge of neural networks and the inherent predictability of the memory accesses performed during inference we can design security measures that work with tensors or chunks of tensors as opposed to single data words.

The bulk of the memory accesses performed in a system enhanced with a machine learning accelerator is made out of reading and writing tensors or chunks of them depending on the their size. We call the integrally accessed chunks of memory "Large Securable Objects"(LSOs). The key property of these chunks is the guarantee that once we touch a memory address belonging to an LSO the rest of it will be read/written it in its entirety. Additionally, as the tensors have to be defined before the inference starts we know in advance how many such LSOs exit. Protecting predefined blocks of memory greatly reduces the amount of metadata required to provide *confidentiality* and *integrity*. Furthermore, this raises the possibility of moving the encryption and integrity metadata (e.g., AES counters and hashes) from the main memory to the System-on-Chip (SoC) thus possibly eliminating the main memory usage of the added security. The security metadata is no longer proportional to the size of the tensors but to the number of tensors. In an ideal scenario, where we have few and continuous LSOs, the security metadata can be stored inside the SoC, removing any extra overhead introduced in memory from previous encryption approaches. In the typical scenario, the security metadata on the memory side is significantly reduced, with the amount of reduction being proportional to the number of LSOs.

References

[1] V. Costan and S. Devadas, "Intel sgx explained," *Cryptology ePrint Archive*, 2016.

[2] D. Lee, D. Kohlbrenner, S. Shinde, K. Asanović, and D. Song, "Keystone: An open framework for architecting trusted execution environments," in *Proceedings of the Fifteenth European Conference on Computer Systems*, 2020, pp. 1–16.

[3] M. Taassori, A. Shafiee, and R. Balasubramonian, "Vault: Reducing paging overheads in sgx with efficient integrity verification structures," in *Proceedings of the Twenty-Third International Conference on Architectural Support for Programming Languages and Operating Systems*, 2018, pp. 665–678.