A Fine-Grained Dynamic Partitioning Against Cache-Based Timing Attacks via Cache Locking

Contacts: **Jeremy GUILLAUME, Vianney LAPOTRE**, firstname.lastname@univ-ubs.fr

Context and motivation

- **Cache-based timing side-channel attacks** : An attacker can deduces secrets with cache contention based attacks such as PRIME+PROBE.
- **Hardware based countermeasures** : There are two main strategies: *randomization* and *partitioning*
  - **Randomization** : consists in randomizing the address indexing in the cache to prevent the attacker from performing the PROBE attack phase. However, data recovery is still possible with a suitable attack like PRIME+PRUNE+PROBE [1].
  - **Partitioning** : consists in isolating the cache resources used by the victim from the ones used by other processes. *PLcache* [2] has been proposed as a lightweight partitioning mechanism. However, it has been demonstrated that this solution is still vulnerable to cache attacks [3].

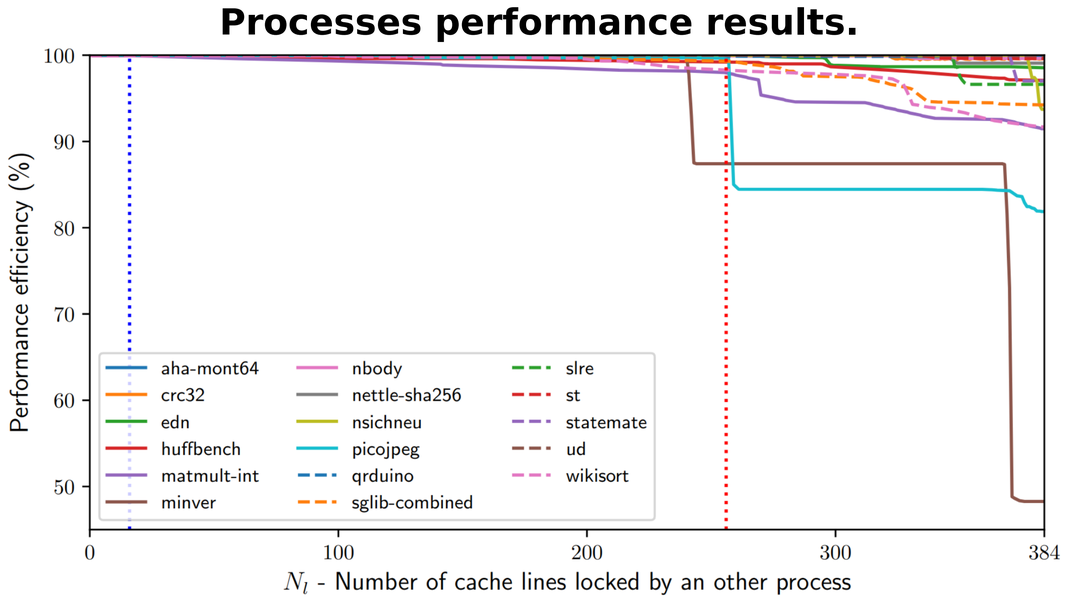
Impact on performance

- **Hardware** : The overall hardware overhead of the solution is lower than 3%.
- **Software** : We observe the performance of a list of processes (the Embench-IoT 1.0 suite) according to the number of caches lines locked by AES-128 (blue dotted line) and Camellia (red dotted line). Even in the worst-case scenario, performance remains close to 90% from initials.

Post-implementation area on Kintex-7 FPGA.

		Cache	CPU
Baseline	LUTs	980	5,661
	FFs	1,065	3,465
	BRAMs	8.5	8.5
Protected	LUTs	1,007 (+2.8%)	5,683 (+0.7%)
	FFs	1,077 (+1.1%)	3,481 (+0.3%)
	BRAMs	8.5	8.5

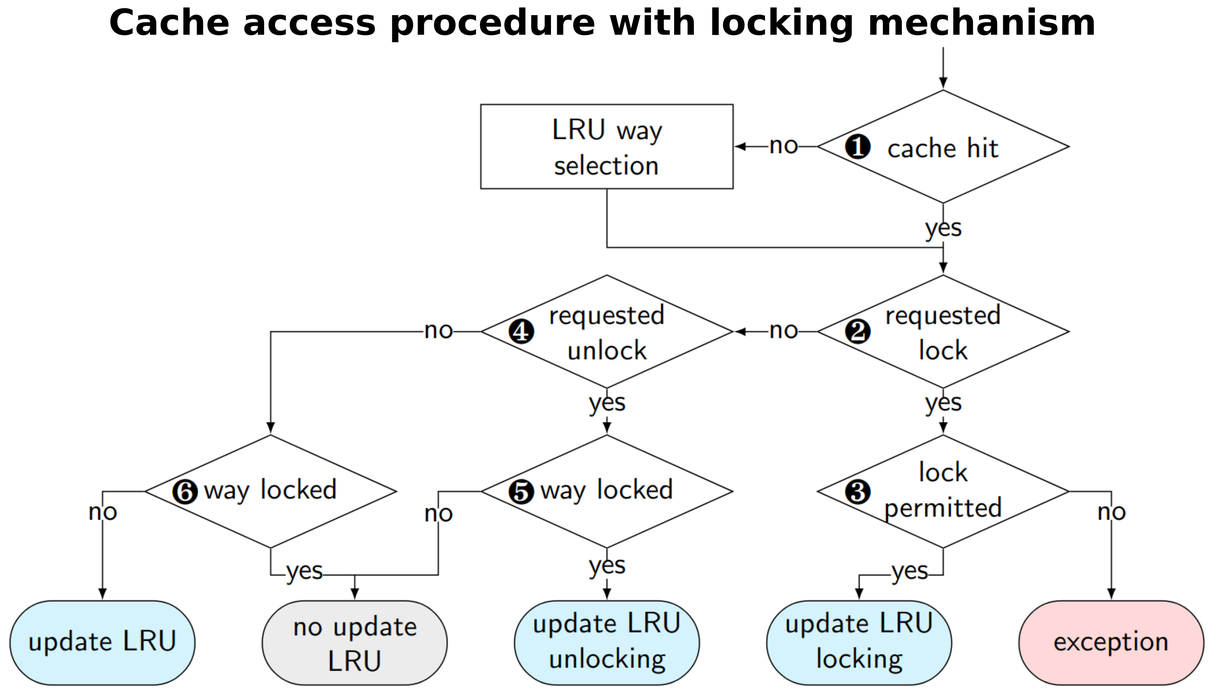
Processes performance results.



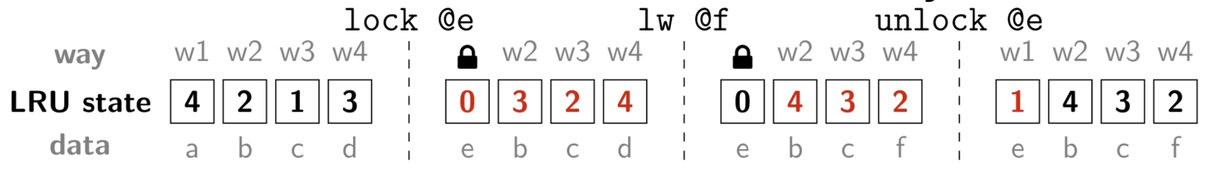
Proposed countermeasure [4]

- **Concept:** we propose a cache partitioning mechanism based on *PLcache*[2] with the addition of a new LRU update mechanism to remove the vulnerabilities from the initial solution.
- **Locking mechanism** : the LRU state is set to 0 when the cache line is locked, and kept unchanged until being unlock. The use-case illustrates the LRU state changes.

Cache access procedure with locking mechanism



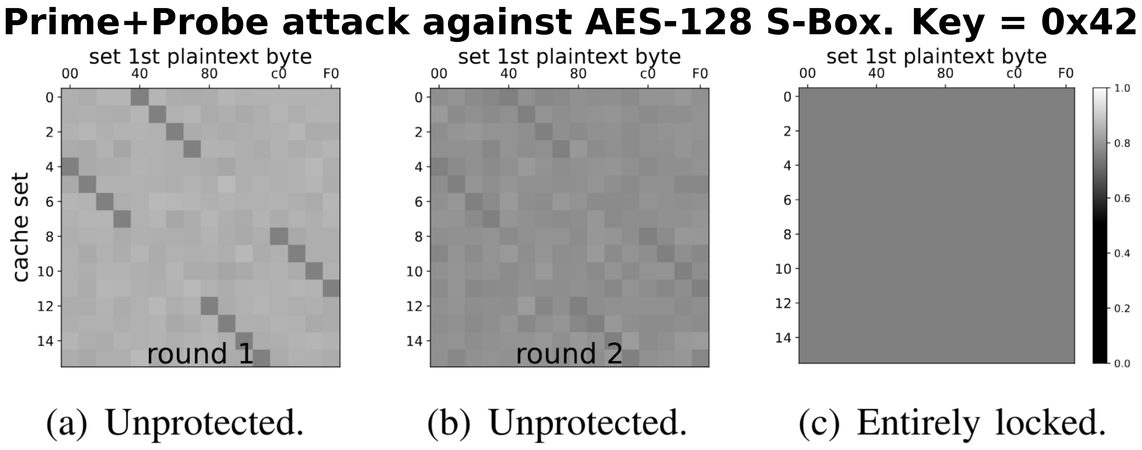
***Use-case:* Behavior of LRU states considering our lock mechanism with N =4 ways**



Impact on security

- The figure shows the results of the PROBE attack phase against the first key byte, (a) & (b) without and (c) with the locking mechanism.
- Results highlight that the attacker cannot observe any difference in time to access the locked data, ensuring security against cache attacks.

Prime+Probe attack against AES-128 S-Box. Key = 0x42.



(a) Unprotected. (b) Unprotected. (c) Entirely locked.

Conclusion and perspectives

- We proposed a fine-grained partitioning relying on a cache locking mechanism to thwart cache-based timing attacks, without significantly impacting processor performance.
- As future works, we aim to study this countermeasure on a more complex processor such as the CVA6, with an OS running on it. Additionnaly, we could investigate the combination of this countermeasure with others, like randomization, to also prevent attacks like Spectre and Meltdown.

[1] Antoon Purnal, Lukas Giner, Daniel Gruss, and Ingrid Verbauwhede. "Systematic Analysis of Randomization-based Protected Cache Architectures". In IEEE Symposium on Security and Privacy (SP). 2021.
[2] Zhenghong Wang and Ruby B Lee. “New cache designs for thwarting software cache-based side channel attacks”. In: Proceedings of the 34th annual international symposium on Computer architecture. 2007.
[3] Nicolas Gaudin et al. “Work in Progress: Thwarting Timing Attacks in Microcontrollers using Fine-grained Hardware Protections”. In: IEEE European Symposium on Security and Privacy Workshops (EuroS&PW). 2023.
[4] Nicolas Gaudin, Pascal Cotret, Guy Gogniat, Vianney Lapôtre. "A Fine-Grained Dynamic Partitioning Against Cache-Based Timing Attacks via Cache Locking". In IEEE Computer Society Annual Symposium on VLSI (ISVLSI). 2024.