Enhancing Keystone Security Against Cache Timing Attacks: A Modular Approach

Oussama Elmnaouri, Pascal Cotret, Vianney Lapotre, Loïc Lagadec

Enhancing Keystone Security Against Cache Timing Attacks: A Modular Approach

Oussama Elmnaouri*, Pascal Cotret*, Vianney Lapôtre†, Loïc Lagadec*
* Lab-STICC, UMR CNRS 6285, ENSTA (29806 Brest Cedex 9, France)
firstname.lastname@ensta.fr
† Lab-STICC, UMR CNRS 6285, Université de Bretagne-Sud (56100 Lorient, France)
vianney.lapotre@univ-ubs.fr

***Abstract*—Confidential computing includes various methods to enhance data security, notably by processing sensitive information within Trusted Execution Environments (TEEs). However, TEEs remain vulnerable to Side-Channel Attacks (SCAs), such as cache timing attacks, which exploit timing variations to extract confidential data. Existing TEE designs do not provide sufficient protection against these threats, highlighting the need for stronger security measures. This study focuses on integrating countermeasures specifically targeting timing and cache vulnerabilities within a TEE. The implementation will leverage the RISC-V architecture to explore its potential in mitigating SCA within TEE.**

***Index Terms*—Computer Architecture, Confidential Computing, Hardware Security, TEEs, SCAs.**

I. Introduction

Trusted Execution Environments (TEEs) are essential for protecting sensitive data by providing isolated environments that guarantee both data confidentiality and integrity. Notable TEEs include proprietary solutions like ARM TrustZone [1], Intel SGX [2] and AMD SEV [3], as well as open source solutions such as Keystone [4] and Penglai [5] for RISC-V. While TEEs offer strong protection against numerous software attacks, they remain vulnerable to cache-based and timing side-channel attacks. These attacks exploit variations in execution time or cache access patterns to extract sensitive data, posing a major challenge to TEE security.

Microarchitectural cache timing attacks, such as Prime+Probe, Flush+Reload, and Evict+Time [6], pose significant challenges to the security of TEEs [7] as they can leak sensitive data by analyzing cache accesses. Various countermeasures have been proposed to mitigate these risks [6], but each comes with trade-offs, some cause high performance overhead, while others do not fully protect against all types of side-channel attacks.

II. Threat Model

The threat model considers that applications executing in both the Rich Execution Environment (REE) and the TEE are vulnerable to cache-based and transient execution side-channel attacks [8]. The attacker is any application running on the system that shares the cache with the victim. This includes both untrusted applications in the REE and potentially malicious applications in the TEE. We do not consider physical attacks (e.g., fault injection, power analysis).

The work presented in this paper was realized in the frame of the SCAMA project number ANR-23-CE39-0011, supported by a grant of the French National Research Agency (ANR).

III. Keystone study

Keystone [4] is an open-source TEE designed for RISC-V processors, combining both security concepts from ARM TrustZone and Intel SGX to establish a clear separation between two execution domains. The non-secure world operates under normal processor conditions, running the untrusted operating system and normal applications while the secure world ensures the execution of sensitive applications, isolated from both the operating system and other applications (sensitive and normal). Keystone is based on a Security Monitor (SM) that manages the entire lifecycle of enclaves and ensures a secure communication between the two worlds by using the RISC-V Physical Memory Protection (PMP) [9] to enforce memory isolation, ensuring that confidential data and enclaves are safeguarded against unauthorized access, providing a flexible and efficient security framework. Figure 1 presents the architecture of a secure system based on enclaves, highlighting the different privilege levels and the separation between trusted and untrusted worlds.

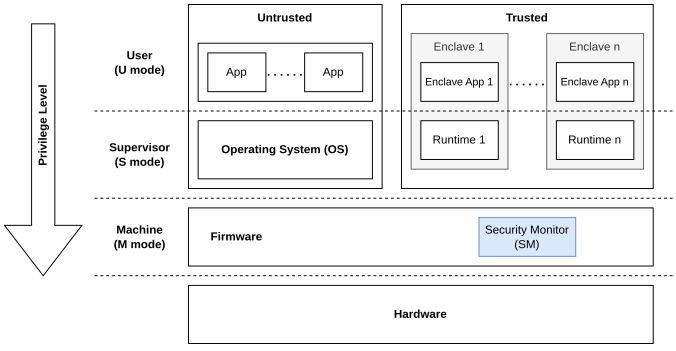

Fig. 1. Keystone Architecture: Secure and Non-Secure World Isolation (adapted from [4]).

IV. Countermeasures on Keystone

Various countermeasures have been developed to mitigate cache timing side-channel attacks [6], each addressing different vulnerabilities:

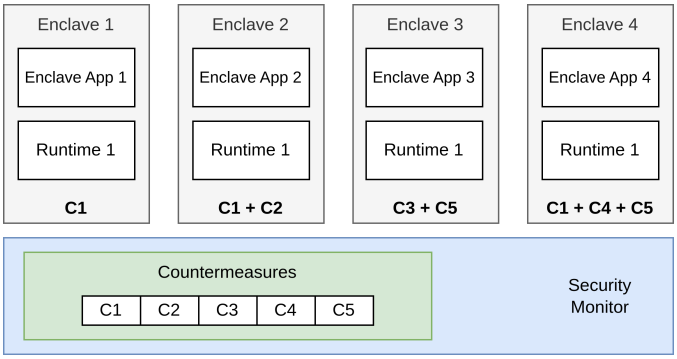

Fig. 2. Flexible Enclaves with Customizable Countermeasures library.

- **Constant-time execution:** Ensures that both cache access patterns and control flow remain independent of secret data, preventing any information leakage through cache timing attacks. This technique is widely used in cryptographic algorithms like AES to prevent key leakage due to execution time variations [10].
- **Noise injection:** Introduces randomness in timing measurements or accesses to shared resources to obscure variations exploitable by an attacker, thereby preventing the leakage of sensitive information. This technique is widely used in cryptographic implementations, real-time systems, and secure embedded devices [11], [12].
- **Enforcing Determinism:** Eliminates execution time variations that could be exploited by an attacker to extract sensitive information through timing channels. This technique is widely integrated in debugging frameworks, cloud computing, and virtual machines security [13], [14].
- **Time Partitioning:** Controls access to shared resources over time to prevent cache timing-based attacks. This is achieved through techniques that influence concurrent resource access and program transitions. This approach is widely used in countermeasures such as Cache Flushing [15], Lattice Scheduling [16], and Execution leases [17].
- **Hardware Partitioning:** Isolates hardware resources to prevent cache side-channel attacks by ensuring that each process has its dedicated space. This technique is widely used in countermeasures such as cache locking [18], Cache Coloring [19], and Quasi-Partitioning [20].

This work aims to develop a flexible and modular framework within Keystone. Although existing research primarily addresses individual mitigation techniques, our approach integrates these countermeasures into a unified framework, thereby providing users with enhanced flexibility to selectively enable the protections that best align with their specific security and performance requirements. One possible direction is to integrate these protections within the SM to enhance enclave lifecycle management and security [21]. However, this remains an open question, and further analysis is required to assess the feasibility, trade-offs, and effectiveness of such an approach. Figure 2 illustrates how we envision the Keystone structure with the integration of these security techniques and protections.

REFERENCES

[1] B. Ngabonziza, D. Martin, A. Bailey, H. Cho, and S. Martin, "Trustzone explained: Architectural features and use cases," in *CIC*, pp. 445–451, 2016.

[2] V. Costan and S. Devadas, "Intel SGX explained." Cryptology ePrint Archive, Paper 2016/086, 2016.

[3] I. Advanced Micro Devices, "Amd sev-snp: Strengthening vm isolation with integrity protection and more," tech. rep., Advanced Micro Devices, Inc., January 2020. White Paper.

[4] D. Lee, D. Kohlbrenner, S. Shinde, K. Asanović, and D. Song, "Keystone: an open framework for architecting trusted execution environments," in *EuroSys*, EuroSys '20, (New York, NY, USA), Association for Computing Machinery, 2020.

[5] E. Feng, X. Lu, D. Du, B. Yang, X. Jiang, Y. Xia, B. Zang, and H. Chen, "Scalable memory protection in the PENGLAI enclave," in *USENIX OSDI*, pp. 275–294, USENIX Association, July 2021.

[6] Q. Ge, Y. Yarom, D. Cock, and G. Heiser, "A survey of microarchitectural timing attacks and countermeasures on contemporary hardware," *Journal of Cryptographic Engineering*, vol. 8, pp. 1–27, 04 2018.

[7] M. Ghaniyoun, K. Barber, Y. Xiao, Y. Zhang, and R. Teodorescu, "Teesec: Pre-silicon vulnerability discovery for trusted execution environments," in *ISCA*, ISCA '23, (New York, NY, USA), Association for Computing Machinery, 2023.

[8] W. Wang, "Side channel risks in hardware trusted execution environments (tees)," in *Side Channel Risks in Hardware Trusted Execution Environments (TEEs)*, May 2019. Presented at a research event.

[9] R.-V. International, *The RISC-V Instruction Set Manual: Volume II: Privileged Architecture, Version 20240411, Apr. 2024, Section 3.7.*, April 2024.

[10] D. A. Osvik, A. Shamir, and E. Tromer, "Cache attacks and countermeasures: the case of aes," in *CT-RSA*, CT-RSA'06, (Berlin, Heidelberg), p. 1–20, Springer-Verlag, 2006.

[11] W.-M. Hu, "Reducing timing channels with fuzzy time," in *RISP*, pp. 8–20, 1991.

[12] E. Brickell, "Technologies to improve platform security," in *CHES - Invited Talk*, (Nara, Japan), Intel Corporation, September 2011.

[13] W. Wu, E. Zhai, D. Jackowitz, D. Wolinsky, L. Gu, and B. Ford, "Warding off timing attacks in deterland," *ArXiv*, vol. abs/1504.07070, 2015.

[14] A. Aviram, S.-C. Weng, S. Hu, and B. Ford, "Efficient system-enforced deterministic parallelism," *Commun. ACM*, vol. 55, p. 111–119, May 2012.

[15] Y. Zhang and M. K. Reiter, "Düppel: retrofitting commodity operating systems to mitigate cache side channels in the cloud," in *CCS*, CCS '13, (New York, NY, USA), p. 827–838, Association for Computing Machinery, 2013.

[16] D. E. Denning, "A lattice model of secure information flow," *Commun. ACM*, vol. 19, p. 236–243, May 1976.

[17] M. Tiwari, X. Li, H. M. G. Wassel, F. T. Chong, and T. Sherwood, "Execution leases: A hardware-supported mechanism for enforcing strong non-interference," in *2009 42nd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pp. 493–504, 2009.

[18] Z. Wang and R. B. Lee, "New cache designs for thwarting software cache-based side channel attacks," *SIGARCH Comput. Archit. News*, vol. 35, p. 494–505, June 2007.

[19] T. Kim, M. Peinado, and G. Mainar-Ruiz, "STEALTHMEM: System-Level protection against Cache-Based side channel attacks in the cloud," in *USENIX Security*, (Bellevue, WA), pp. 189–204, USENIX Association, Aug. 2012.

[20] Z. Zhou, M. K. Reiter, and Y. Zhang, "A software approach to defeating side channels in last-level caches," in *CCS*, CCS '16, (New York, NY, USA), p. 871–882, Association for Computing Machinery, 2016.

[21] S. Nashimoto, R. Ueno, and N. Homma, "Comparative analysis and implementation of jump address masking for preventing tee bypassing fault attacks," in *ARES*, ARES '24, (New York, NY, USA), Association for Computing Machinery, 2024.