

μLEAK: Bypassing MPU Isolation on Cortex-M7 via Cache-Timing Attacks

Muhammad Hammad Bashir¹, Taegyu Kim¹, Arslan Khan¹ and Kyungtae Kim²

¹Pennsylvania State University

²Dartmouth College

Abstract

High-performance microcontrollers (MCUs) rely on hardware L1 caches to mask the high latency of embedded Flash memory, inadvertently introducing data-dependent timing channels. We present μLEAK, an unprivileged Prime+Probe attack on ARM Cortex-M7 that recovers AES keys despite MPU-enforced privilege separation. We show that vendor-recommended mitigations eliminate leakage but can reduce AES throughput by up to 73%, highlighting a security-performance trade-off driven by NVM-constrained execution.

Introduction

Modern microcontrollers (MCUs), particularly high-performance variants based on the ARM Cortex-M7 architecture, are increasingly constrained by the latency of on-chip non-volatile memory (NVM). While CPU core frequencies have scaled into the hundreds of megahertz, embedded Flash access latency has not scaled proportionally.

To bridge this processor-memory performance gap and sustain throughput, Cortex-M7 MCUs incorporate performance mechanisms traditionally associated with higher-end systems, including tightly coupled memories (TCM) [1], L1 instruction and data caches, and vendor-specific accelerators (e.g., ART)[2]. Although these caches are smaller and architecturally simpler than those in application-class processors, they still introduce data-dependent timing variability and thereby expand the microarchitectural side-channel surface. Table 1 summarizes representative Cortex-M7 MCUs from major vendors that include L1 I/D caches.

Prior work has often modeled MCUs as largely deterministic platforms, an assumption that is increasingly misaligned with high-performance MCU designs. For example, BUSTED [3] shows that even in embedded systems, contention on shared microarchitectural resources can induce exploitable timing effects (e.g., bus contention). On Cortex-M7 MCUs, L1 I/D caches further introduce data-dependent timing variation, weakening the determinism premise that has historically shaped how MCU side channels are treated.

Existing work has mainly focused on cache side channels on application-class processors (e.g., Intel x86, ARM Cortex-A). However, cache-based attacks and mitigation trade-offs in MCUs remain comparatively underexplored. This is particularly concerning given the deployment of Cortex-M7 MCUs in security-critical IoT, automotive, and industrial control systems.

In this work, we explore sources of side-channels in

modern MCUs. As a Proof of Concept, we demonstrate a Prime+Probe attack on a Cortex-M7 MCU that recovers full AES keys, despite MPU-enforced privilege separation. We further evaluate vendor-recommended mitigations [4] and find that disabling the L1 I/D caches eliminates the leakage but incurs up to a 73% performance penalty, highlighting the trade-off between real-time performance and microarchitectural security.

Threat Model

We consider a restrictive threat model in which an adversary compromises an unprivileged task that executes on the same Cortex-M7 core as a privileged victim and shares the L1 cache. We assume the MPU is correctly configured and enforced, preventing the attacker from directly accessing the victim’s memory region or any secret-key material. Because the attacker runs unprivileged, it cannot access or configure the system control space (e.g., DWT). We assume the RTOS provides read-only access to a preconfigured TIM1, which the attacker uses as a timing source for its analysis.

Attack Design & Implementation

We implement a Prime+Probe cache-timing attack [5] against the AES T-table implementation in mbedTLS (v2.28.10) [6] on an STM32F769I-EVAL (Cortex-M7) [2], which features a 16KB, 4-way set-associative L1 D-cache with pseudo-random replacement policy.

A critical challenge is the cache replacement policy, where standard eviction ($N = 4$) yields probabilistic failures. We mitigate this via *Iterative Priming* to maximize eviction probability.

Online Analysis: For each trace, we scan the 32 cache sets corresponding to one T-table, time the probe using TIM1, and transmit $\langle \text{pt}, \text{iter}, \text{vector} \rangle$ over UART after the measurement window.

Offline Phase: From the collected traces, we recover

Table 1: Representative Cortex-M7 MCU families that include the core’s dedicated L1 instruction/data caches (per core; varies by part).

Vendor	MCU families	L1 I/D (KB)	Typical segment
STMicroelectronics	STM32F7; STM32H7	16/16; 16/16–32/32 [†]	General-purpose HP MCUs
NXP	i.MX RT10xx; S32K3xx	part-dependent [†]	Real Time; automotive (S32K3)
Microchip	SAM E70/S70/V70/V71; SAM S70	16/16	Industrial, general-purpose MCUs
Infineon	TRAVEO T2G (CYT4xx); XMC7000	16/16	Automotive / industrial MCUs

[†]Exact cache size/configuration depends on the specific subfamily/part.

each key byte by scoring all hypotheses with a Difference-of-Means (DoM) over probe latencies relative to a noise baseline. We normalize scores into Z-scores, and the correct key consistently produces the dominant peak.

Evaluation

Despite strict MPU isolation, we recovered the full 128-bit key using **7,680 traces** per byte. Figure 1 confirms high distinguishability, with the correct hypothesis reaching **6.28 σ** confidence¹.

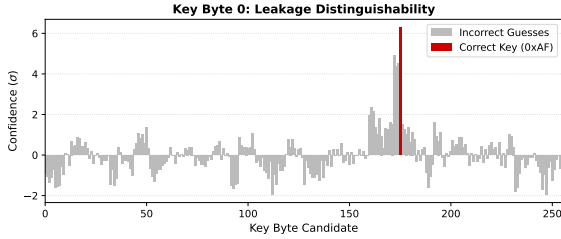


Figure 1: Key recovery for Byte 0 (0xAF).

Discussion & Mitigations

Tightly Coupled Memory (TCM). Many Cortex-M7 MCUs integrate ITCM and DTCM to provide low-latency storage for time-critical code and data [1]. Placing AES lookup tables in TCM bypasses the L1 data cache and removes cache-mediated leakage. However, TCM placement alone does not guarantee constant-time execution and can still introduce timing variability if the layout is not carefully controlled [7].

MPU-based non-cacheable tables. A vendor-recommended mitigation is to use the MPU to map the table region as *Normal, non-cacheable*, preventing caching and eliminating Prime+Probe leakage through the data cache [4]. While this eliminates cache-based timing variability, it incurs a prohibitive performance penalty, i.e., 73% performance degradation in AES throughput when this mitigation is applied, which negates the performance benefits of caches.

Hardware acceleration. When available, offloading AES to a hardware crypto peripheral removes software T-table lookups. However, crypto/hash acceleration

is SKU-dependent (even within closely related MCU lines), and cannot be assumed across Cortex-M7 deployments [2].

The NVM Latency Gap. The vulnerability stems from the CPU–Flash latency mismatch that necessitates aggressive caching. Emerging embedded NVMs (e.g., STT-MRAM, RRAM) may narrow this gap, but it is unclear whether they will replace flash at scale. Even with faster NVM, caches may remain for bandwidth and performance, so the more realistic impact is enabling sensitive code and data to avoid cacheable paths.

Conclusion and Future Work

We demonstrate practical Prime+Probe caching timing leakage on a Cortex-M7 (STM32F769I-EVAL) despite MPU-based privilege separation. Future work will broaden the evaluation across vendors and study Armv8-M TrustZone to test whether Secure/Non-secure partitioning provides stronger microarchitectural isolation than the MPU model.

References

- [1] Arm Ltd. *ARM Cortex-M7 Processor*. <https://developer.arm.com/Processors/Cortex-M7>. Accessed: 2025-12-13. 2025.
- [2] STMicroelectronics. *STM32F769I-EVAL Evaluation Board*. <https://www.st.com/en/evaluation-tools/stm32f769i-eval.html>. Accessed: 2025-12-13. 2025.
- [3] Cristiano Rodrigues, Daniel Oliveira, and Sandro Pinto. “BUSTed!!! Microarchitectural Side-Channel Attacks on the MCU Bus Interconnect”. In: *2024 IEEE Symposium on Security and Privacy (SP)*. 2024, pp. 3679–3696. DOI: 10.1109/SP54263.2024.00062.
- [4] STMicroelectronics. *Secure Usage of the Cryptographic Library*. https://wiki.st.com/stm32mcu/wiki/Security:Secure_usage_of_the_Cryptographic_Library. Accessed: 2025-12-13. 2025.
- [5] Dag Arne Osvik, Adi Shamir, and Eran Tromer. “Cache Attacks and Countermeasures: The Case of AES”. In: *Journal of Cryptology* 23.1 (2010), pp. 37–71.
- [6] Mbed TLS Contributors. *Mbed TLS: Open Source Cryptographic Library*. <https://github.com/Mbed-TLS/mbedtls>. Version 2.28.10, accessed: 2025-12-13. 2023.
- [7] Alessandro Barenghi et al. “Exploring Cortex-M Microarchitectural Side Channel Information Leakage”. In: *IEEE Access* 9 (2021), pp. 156507–156527. DOI: 10.1109/ACCESS.2021.3124761.

¹ Reproduction artifacts will be released upon completion of the work.