# Vulnerability Analysis using MASTIK on AES & RSA Implementations on Intel & AMD Systems, with add. tests on hypervisor Systems (Virtualbox)

**Nimish Pandey - s4022297** [1] **Mehul Upase - s4017633** [1] **Divyanshi Singh - s4374827** [1] **Snehil Dey - s4247655** [1]

## Abstract

This report discusses the feasibility and impact of micro-architectural side-channel attacks against cryptographic algorithms, specifically AES and RSA. The work leverages the MASTIK toolkit while studying how to leverage the Prime+Probe technique to exploit cache-based side-channel vulnerabilities on both Intel and AMD systems. Various AES encryption experiments were performed on Intel systems to study the leakage of L1 and L3 caches, creating detailed heatmaps and memory access patterns. The project uniquely extends previous work by validating MASTIK's functionality on AMD processors, showing that it is capable of accessing L1 cache despite limited support for AMD architectures. RSA encryption experiments were only performed on Intel systems, which gave insight into the memory access patterns that may be vulnerable to cache-based attacks. This paper underlines the importance of cache leakage as a threat vector and points to the challenges and limitations of using SCAs across diverse processor architectures, thus providing basic insights into attack mechanisms and implications.

## Statement of Academic Integrity:

We hereby confirm that this report was fully produced by the team members Nimish, Divyanshi, Snehil, and Mehul. We are jointly and severally responsible for all content presented in this work. Sources were attributed wherever used.

## 1. Introduction

Micro-architectural SCAs rely on the complex interaction of software and hardware to leak sensitive information from shared resources such as CPU caches, memory buses, and branch predictors. These attacks do not infringe on the logical security of the software but rather exploit an unintended leakage from the bottom layer hardware. These vulnerabilities are thus intrinsic in processor designs and have therefore become a big challenge to cryptographic systems in modern times.

One of the major attack vectors in SCAs is CPU cache. Caches, being used to store frequently accessed data to optimize the performance of the processor, are shared resources that inherently leak access patterns when concurrently used by multiple processes. The Prime+Probe technique, for example, uses the measurement of cache line use to deduce memory access without requiring shared memory access (Tromer et al., 2010). Similarly, the Flush+Reload attack, observing shared memory regions for cache access patterns, showed high-precision leaks in the AES key (Yarom and Falkner, 2014). These techniques illustrate how the behavior of a cache can be used to deduce cryptographic secrets.

Cryptographic algorithms such as AES and RSA are, of course, instrumental in digital data security. Implementations of these algorithms are, however, subject to SCAs due to the execution time difference or deviation in access to memory. Cache timing attacks, for example, have been able to recover AES encryption keys by analyzing cache hits and misses during encryption processes. Patterns of cache access have also been found to leak key information even in RSA implementations during modular exponentiation Steak.(Ge et al., 2018).

MASTIK is an open-source framework providing the software needed to mount and analyze these types of attacks easily. It provides support for multiple well-known techniques like Prime+Probe and Flush+Reload. They all include implementations for Intel x86-64 processors (Yarom, 2018). It is an excellent tool for any researcher looking to understand and illustrate the possibility of micro-architectural SCAs.

This project extends previous work, using the MASTIK toolkit to investigate side-channel attack opportunities in both AES and RSA encryption. Some studies have thus far concentrated on Intel processors; hence, this work will investigate how this toolkit operates on an AMD system and its ability to reach L1 cache for vulnerabilities. It also researches memory access patterns and cache leakage of RSA encryption on Intel platforms, casting new light on the applicability of SCAs in different hardware settings. (Ge et al., 2018).

This project contributes to the deeper understanding of micro-architectural vulnerabilities through the analysis of

cache access patterns and the effectiveness of SCAs across different processors. The outcome will drive the design of secure cryptographic implementations and give reason to the critical need for mitigation strategies such as Intel's Cache Allocation Technology.(Ge et al., 2018).

## 2. Methodology:

In this regard, the experimental methodology was carefully designed, taking advantage of the MASTIK toolkit, which can perform the Prime+Probe technique to carry out an extensive investigation of micro-architectural SCAs. The proposed methodology allows experiments that are reproducible and easy to understand, thus providing deep insight into cache leakage in cryptographic implementations. The whole process was systematically performed by setting up the experimental environment, carrying out the attack, and analyzing the data collected.

### 2.1. System Setup and Configuration

The experimental setup contained three different systems: one Intel laptop, one AMD running Linux via VirtualBox, and an AMD EPYC server running some variant of the Linux operating system. MASTIK was installed and configured on each system for cache-based SCA experimentation. Although MASTIK is optimized for the Intel x86-64 architecture, attempts have been made to extend the functionality to cover the AMD systems concerning the evaluation of limitations and the capabilities of this tool in accessing L1 cache data. The configuration of systems involved in ensuring that minimum noise and interference are present since this is critical in achieving reliable cache timing measurements.

On the Intel platform, the access patterns for the L1 and L3 cache were studied, but on the AMD systems, due to the architectural limitation of MASTIK, only the L1 cache data were accessible. The cryptographic workloads used are AES encryption on all systems and RSA encryption on the Intel platform only. These workloads were chosen because of their ability to study how cryptographic algorithms interact with the processor caches and find out if there is any leakage.

### 2.2. Prime+Probe Execution

The Prime+Probe technique, as applied in monitoring the usage of the cache line and deducing memory access, forms a focal point of this study in that it can operate without the need for shared memory, especially in situations where either party is uncooperative (Tromer et al., 2010). The basic idea of this type includes populating the cache with an attacker-controlled element in a so-called 'Prime' phase while the probe stage measures cache response times by the cache from which the perpetrator would be able to infer the activities done by the victim process. The two-phase approach will enable a detailed analysis of the cache behavior and memory access patterns in cryptographic operations.

In the AES encryption on the Intel platform, Prime+Probe targeted L1 and L3 caches. The collected dataset consisted of timing measurements with access patterns, visualized as heatmaps and memory maps to bring out the activity occurring in the cache. In the case of AMD, where MASTIK could access only L1 cache, experiments focused on observing whether observable patterns could be detected despite limitations.

### 2.3. Data Collection and Analysis

Cache timing data is collected at high resolution to capture the subtlety of variation in access patterns during experiments. The usage of the cache in AES experiments is visualized by memory maps and heatmaps. In fact, these visualizations helped a lot in finding some cache line contention that pointed toward possible side-channel vulnerabilities. Complementary information about cross-core cache use was extracted with the L3 cache data available on the Intel platform. Because of the MASTIK limitations, only L1 cache activity was able to be analyzed on AMD systems.

In the RSA experiments on the Intel platform, modular exponentiation due to cache access patterns was traced using Prime+Probe. The reason behind these experiments is the search for a pattern that may leak information depending on the key. As MASTIK focuses on architectural differences, the current version does not extend RSA experiments to any platforms.

### 2.4. Challenges and Limitations

While the experiments on leakage in Intel systems were successful, limitations in the support of MASTIK impeded the execution of AMD systems. This had a negative impact and limited the scope of performing an analysis on such a platform. In the case of the AMDv, whose environment was virtualized, there was added noise during the measurements, affecting precision and thus weakening the results. These three limitations evidence the need for further development within tools like MASTIK to respond to different processor architectures in depth.

## 3. Results:

Experiments performed in this work have targeted two vulnerabilities of cryptographic cache-based implementations via the Prime+Probe attack technique. The results of the experiments are categorized mainly into two sections: analysis of AES encryption on an Intel system and raw monitoring of CPU caches across both Intel and AMD systems. RSA encryption experiments on Intel systems were more centered on the activities in L1 cache during modular exponentiation.

### 3.1. AES Encryption on Intel Systems

AES encryption conducted on an Intel system gave many insights into a possible cache leakage pattern, how heat maps and memory access patterns of cache line contention could occur during the conduct of encryption operations; and how to extract sensitive information using Prime+Probe. Such visualization highlights some unique changes in cache usage correlated to cryptographic operations and proves once more the feasibility of micro-architectural side-channel attacks targeting AES.

**Heatmaps for AES Encryption (Intel L1 Cache):** This heatmap illustrates cache access patterns in the L1 cache during AES encryption on the Intel system, highlighting cache line contention indicative of side-channel vulnerabilities.

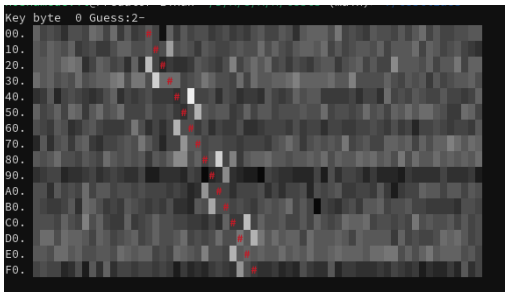


*Figure 1.* Heatmap of Intel L1 cache with AES load

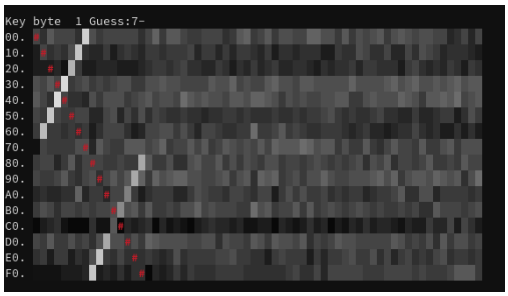


*Figure 2.* Heatmap of Intel L1 cache with AES load

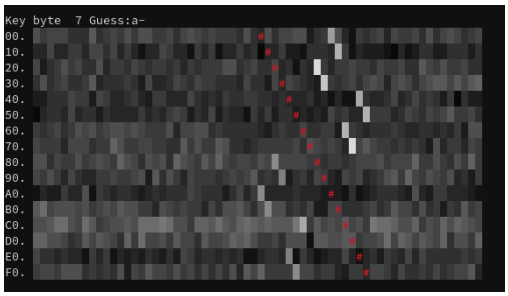


*Figure 3.* Heatmap of Intel L1 cache with AES load

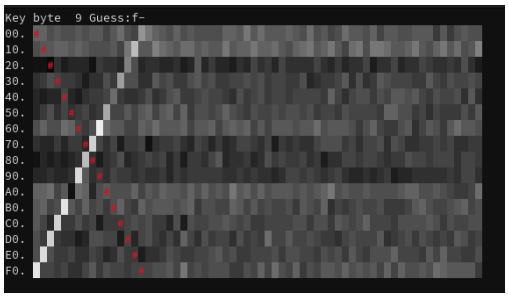


*Figure 4.* Heatmap of Intel L1 cache with AES load

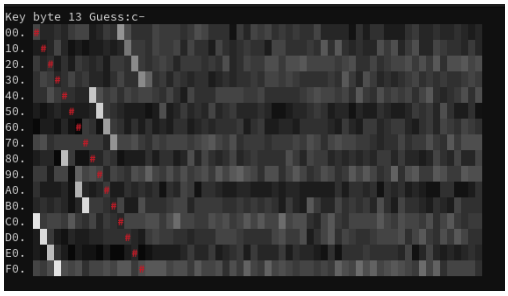


*Figure 5.* Heatmap of Intel L1 cache with AES load

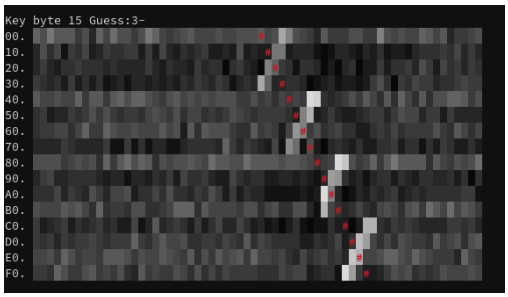


*Figure 6.* Heatmap of Intel L1 cache with AES load

**Heatmaps for AES Encryption (Intel L1 Cache):** This memory map depicts cache usage across L1 cache lines during AES encryption, showcasing patterns that align with cryptographic operations.

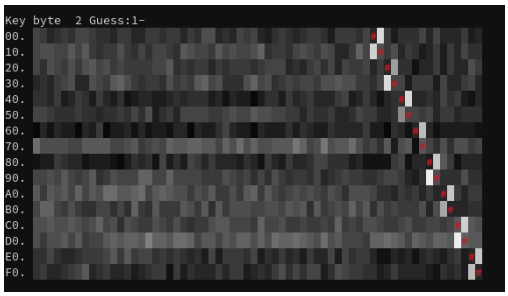


*Figure 7.* Heatmap of Intel L1 cache with AES load

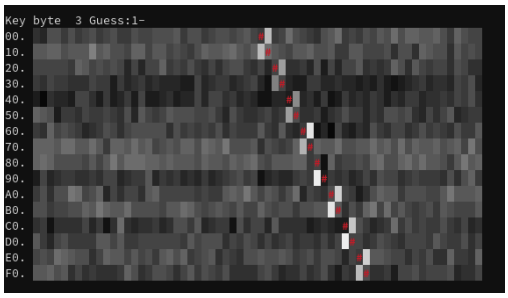


*Figure 8.* Heatmap of Intel L1 cache with AES load

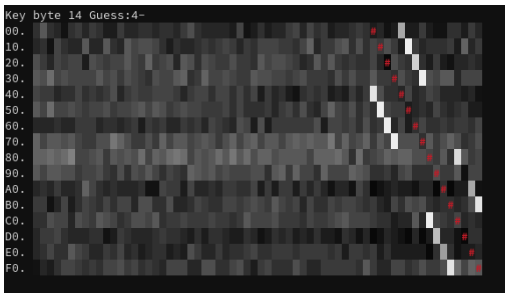


*Figure 9.* Heatmap of Intel L1 cache with AES load

### 3.2. Raw CPU Cache Monitoring

The baseline cache behavior was monitored in raw CPU cache on Intel, AMD, and AMD VirtualBox without any cryptographic operations. In this way, the control results can be compared with the scenarios of cryptographic workload.

On Intel systems, the raw monitoring data captured the cache behavior in L1 and L3 caches, which served as a reference for normal access patterns. In contrast, only L1 cache access was analyzed on AMD systems due to MASTIK's architectural limitation. Despite this limitation, the results depicted observable cache activity on AMD processors, proving that MASTIK is partially functional on non-Intel platforms.

#### 3.2.1. L1 Cache Monitoring on Intel System (No Algorithm):

Cache activity in the L1 cache during raw monitoring on the Intel system, without any cryptographic algorithms running.

**With & without load on Intel System L1 Cache:**

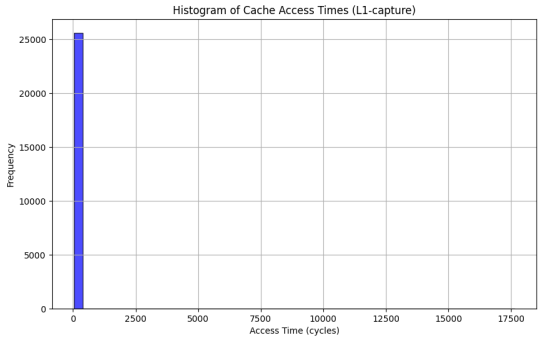


*Figure 10.* Graph of Intel L1 cache with load

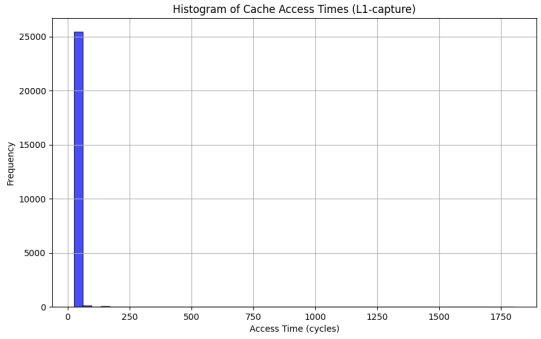


*Figure 11.* Graph of Intel L1 cache without load

**With & without load on Intel System L1 Cache:**

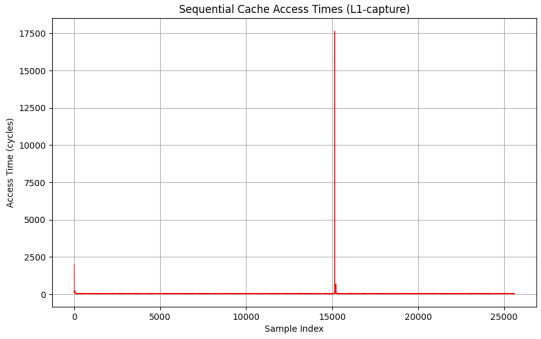


*Figure 12.* Graph of Intel L1 cache with load

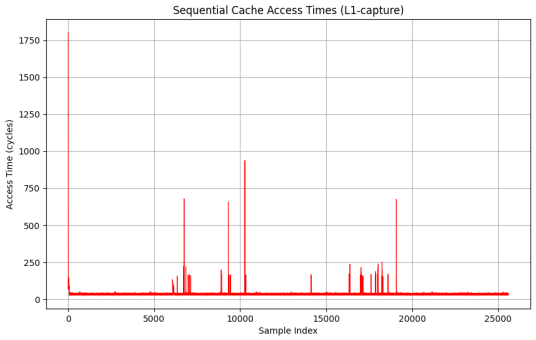


*Figure 13.* Graph of Intel L1 cache without load

**L3 Cache Monitoring on Intel System (No Algorithm):** Cache activity in the L3 cache during raw monitoring on the Intel system, without any cryptographic algorithms running.

**With & without load on Intel System L3 Cache::**

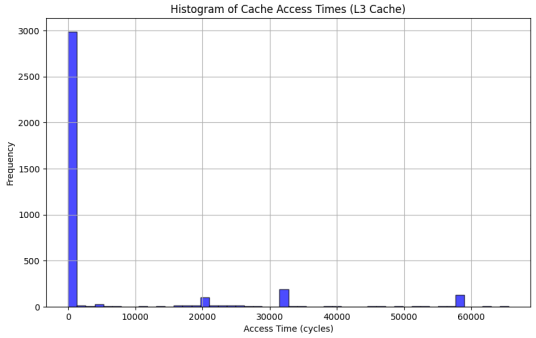


*Figure 14.* Graph of Intel L3 cache with load

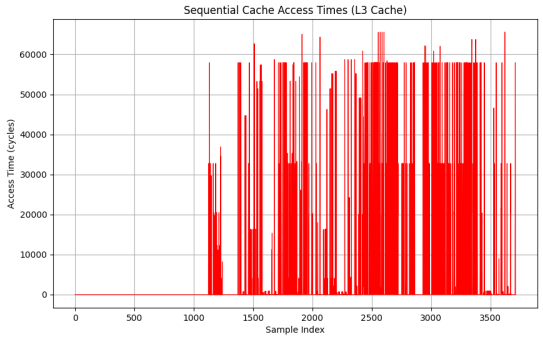


*Figure 15.* Graph of Intel L3 cache with load

**L1 Cache Monitoring on AMD Server (No Algorithm):** Raw monitoring data showing L1 cache activity on the AMD Server, demonstrating observable patterns despite limited tool support.

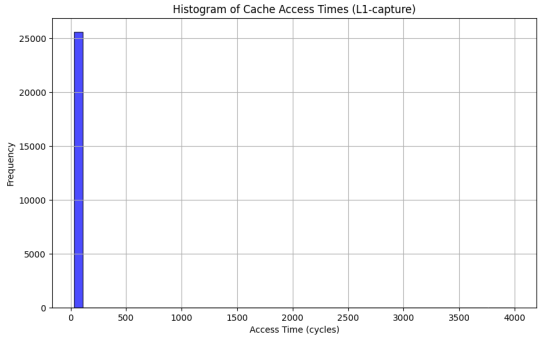


*Figure 16.* Graph of AMD L1 cache with load

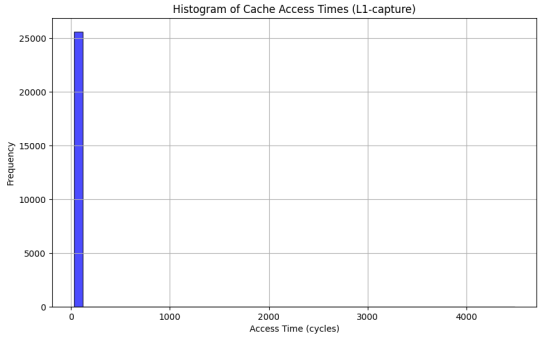


*Figure 17.* Graph of AMD L1 cache without load

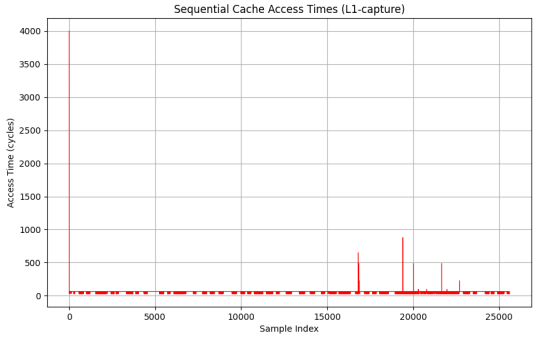


*Figure 18.* Graph of AMD L1 cache with load

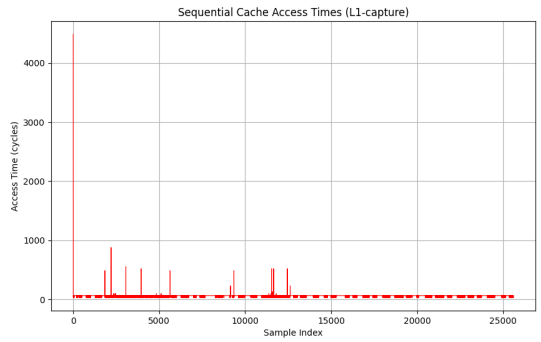


*Figure 19.* Graph of AMD L1 cache without load

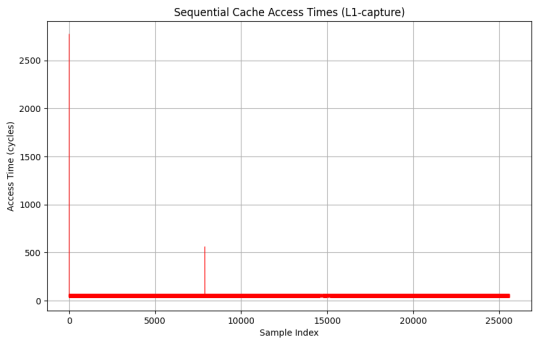


*Figure 22.* Graph of AMDv L1 cache with load

**L1 Cache Monitoring on AMD VirtualBox System (No Algorithm):** Cache activity observed in the L1 cache of the AMD VirtualBox system during raw monitoring.

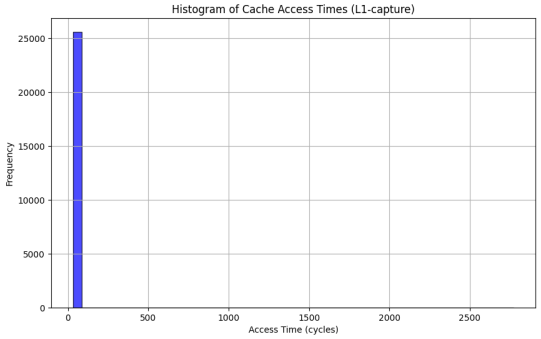


*Figure 20.* Graph of AMDv L1 cache with load

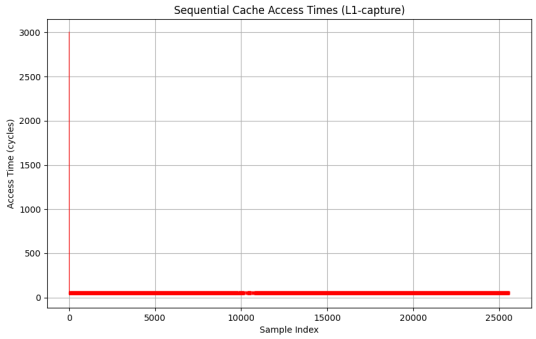


*Figure 23.* Graph of AMDv L1 cache without load

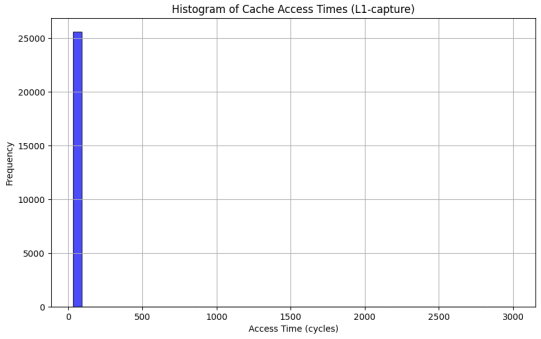


*Figure 21.* Graph of AMDv L1 cache without load

### 3.3. RSA Encryption on Intel Systems

Various experiments involving RSA encryption were implemented using an Intel-based system, such as modular exponentiation behavior and effects induced by the L1 cache. Employing the technique called Prime+Probe to measure cache activity would certainly help uncover specific patterns leading to either the execution of an RSA algorithm or its potential disclosure due to side-channel leaks.

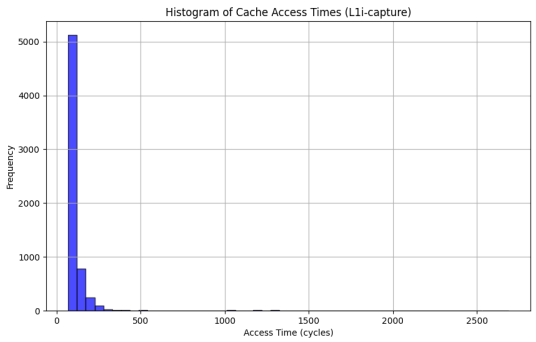


*Figure 24.* L1 Cache Activity During RSA Encryption (Intel System): Access patterns to the L1 cache while doing RSA encryption on the Intel system, possibly leaking information via side channels

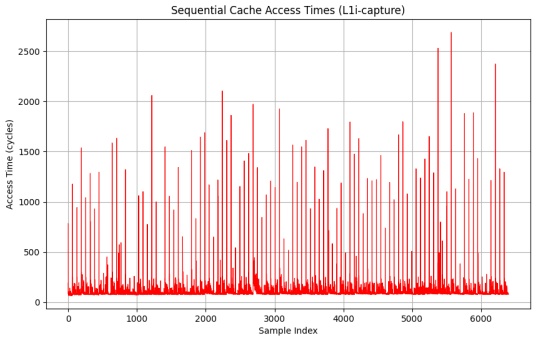


*Figure 25.* Cache-timing measurements during RSA Encryption (Intel System): Timing data for L1 cache accesses during RSA encryption on the Intel system. The variations indicate leakage.

### 3.4. Comparative Observations and Challenges

These results indicate large differences in the cache behavior of Intel and AMD systems, pointing out that the architectural influence is major regarding the feasibility of side-channel attacks. While the AES and RSA experiments on Intel gave quite obvious evidence of cache leakage, in general, the results under AMD pointed out the limited support and functionality of the MASTIK toolkit for non-Intel architectures. These findings foreshadow further development in SCA tools to address broader hardware compatibility.

## 4. Discussion and Conclusion

### 4.1. Discussion

This research work focuses on the vulnerability of modern processors to micro-architectural side-channel attacks, targeting specifically the cryptographic algorithms AES and RSA. We used the Prime+Probe technique with the MASTIK toolkit to investigate cache behaviors that could hint at possible information leakage.

#### 4.1.1. Feasibility of SCAs on AES and RSA

As discussed, clear patterns of cache access while running AES encryption showed clear patterns on our experiments running the Intel systems in both L1 and L3 cache heatmap and access patterns. In that respect, this result does provide validation to various works stating cache-based implementations of AES are vulnerable against side-channel attacks (Osvik et al., 2006). Similarly, for RSA encryption, monitoring L1 cache unveiled variations related to the modular exponentiation process that were consistent with previously reported leakage patterns exploitable for extracting an RSA private key (Alam et al., 2018). Without a goal of key extraction, at least, our observations with this regard do bring into sharp light the risks which possible such cryptographic implementations pose.

#### 4.1.2. Architectural Variations: Intel vs. AMD

Another important aspect of this work is the study of the feasibility of SCAs in various processor architectures. Indeed, on Intel platforms, MASTIK was able to access both the L1 and L3 caches for deep analysis. In AMD systems, it only accesses the L1 cache. Such a limitation of access can be due to their architecture, as well as might be due to a lack of exhaustive documentation regarding AMD's microarchitecture, making the development of attack tools universal (Martinoli et al., 2022). Despite these limitations, the observable L1 cache patterns on AMD processors indicate that they are not resistant to SCAs, though the attack surface is comparatively reduced.

#### 4.1.3. Challenges and Limitations

Several challenges arose during the experiment. First, the virtualized environment on the AMD laptop added extra noise that may reduce the accuracy of the cache timing measurement. Second, due to the limited support of MASTIK for AMD architectures, our analysis had to focus only on L1 cache, whereas higher-level caches might still be vulnerable. These challenges indicate the need for more flexible tools supporting various processor architectures in order to enable comprehensive SCA research.

While our experiments have shown cache leakage patterns, we did not quantify the information leakage, neither did we attempt key recovery. Key extraction methodologies and consequences of these vulnerabilities in real-life applications are left for future studies.

### 4.2. Conclusion

This research demonstrates the vulnerability of AES and RSA cryptographic implementations to cache-based side-channel attacks in modern processors. Using the Prime+Probe technique with the MASTIK toolkit, we show that on Intel systems, cache access can be observed in cryptographic operations, which could be indicative of information leakage. Further work on AMD systems highlighted some of the limitations of current tools and demonstrated a clear need for further development to include more architectures.

These results imply that serious countermeasures will be needed in order to reduce the SCA risks. Some of the techniques which can improve resistance include cache partitioning, randomization, and noise injection. Hardware-based defenses, for instance, Intel's Cache Allocation Technology, is promising, though how these generalize across architectures is an open question (Ge et al., 2018). SCA Paper.

In all, this work contributes to the understanding of microarchitectural vulnerabilities in cryptographic systems and strengthens the demand for continuous effort toward secure implementations resistant against side-channel attacks. Mastering the challenges pointed out here will turn out to be essential for a strategic security improvement of cryptographic applications within different computing environments.

## 5. Resources Used & Links to Code + Video:

- MASTIK Code (Base of Our Work): MASTIK's GitHub Code
- Our Code (RSA Implementation): Group 11 Code
- Link to Video Presentation: Group 11 Video

## Bibliography


S. Alam, D. Mukhopadhyay, and S. Bhattacharya. Winter is here! a decade of cache-based side-channel attacks, detection and mitigation. *HAL Archives*, 2018.

Q. Ge, Y. Yarom, D. Cock, and G. Heiser. A survey of microarchitectural timing attacks and countermeasures on modern processors. *Journal of Cryptographic Engineering*, 8(1):1–27, 2018.

V. Martinoli, Y. Teglia, A. Bouagoun, and R. Leveugle. Cva6's data cache: Structure and behavior. *arXiv preprint arXiv:2202.03749*, 2022.

D. A. Osvik, A. Shamir, and E. Tromer. Cache attacks and countermeasures: The case of aes. *Cryptographers' Track at the RSA Conference*, pages 1–20, 2006.

E. Tromer, A. Shamir, and C. Percival. Cache attacks and countermeasures: The case of aes. In *Cryptographers' Track at the RSA Conference*, pages 1–20. Springer, 2010.

Y. Yarom. Mastik toolkit documentation and resources, 2018. URL `https://cs.adelaide.edu.au/~yval/Mastik/`.

Y. Yarom and K. Falkner. Flush+reload: A high resolution, low noise, l3 cache side-channel attack. *Proceedings of the 23rd USENIX Security Symposium*, pages 719–732, 2014.