Cache Attacks and Countermeasures – the Case of AES

Elena Frank, Frederik Safner & Lukas Kaibel

July 5, 2023

Contents

4 Countermeasures

Several methods can be considered to mitigate information leakage in cryptographic systems, each with unique trade-offs:

Avoiding Table Lookups: Techniques include using logical operations instead of table lookups, using bitslice implementations to execute multiple encryptions simultaneously through vectorization, or placing lookup tables in registers instead of cache. These methods may reduce performance or be architecture-dependent.

Alternative Lookup Tables: By using smaller or combined lookup tables, the chance of a given memory block not being accessed during encryption is reduced. This method is more resistant to synchronous attacks but still vulnerable to asynchronous attacks, and is most effective when used with other countermeasures.

Data-Oblivious Memory Access Pattern: This involves altering memory access patterns to be independent of data. Techniques include reading all entries from relevant tables, memory shuffling or permutation, or adding spurious memory accesses. All these methods result in some performance slowdown.

Algorithmic Masking Techniques: These randomize data-dependent operations. Current AES masking methods are either insecure or significantly slower.

Cache State Normalization and Process Blocking: This method involves normalizing the cache state before and after encryption and is effective against synchronous attacks but not asynchronous ones.

Disabling Cache Sharing or Caching Mechanism: Disabling cache sharing can prevent cache state effects from crossing process boundaries but may require extensive resources. A more drastic measure is disabling the CPU's caching mechanism altogether, which significantly affects performance. A more feasible alternative is the "no-fill" mode, which requires privileged mode.

Dynamic Table Storage: This method involves randomizing table lookup locations or changing table order multiple times during encryption. The effectiveness of this method depends on the architecture.

Hiding Timing Information: This involves adding random delays or normalizing all timings to a fixed value, but it doesn't provide complete protection against certain types of attacks.

Selective Round Protection: This method focuses on protecting specific rounds of encryption while leaving the internal rounds unprotected.

OS-Level Support: Some countermeasures may require OS-level support or privileged operations. One solution could be to provide secure execution of cryptographic primitives as OS services. An alternative is providing a secure execution facility to user processes.

These methods represent various trade-offs between security and performance. Due to the complexity and architecture-dependency, secure implemen-

tation requires a delicate balance and no one-size-fits-all solution exists.

5 Conclusion