Analysis of Time-Driven Cache Attacks on Block Ciphers

ABSTRACT

Most cryptographic ciphers in use today are vulnerable to a class of attacks known as side-channel attacks. These attacks utilize information leakage through unintentional covert channels like time, power consumption, and electro-magnetic radiation of the device executing the cipher. For block ciphers, cache memories present in modern microprocessors play a crucial role in abetting side-channel leakage. Attacks that make use of leakage caused from cache memories are called *cache-attacks*. This thesis analyzes a category of cache-attacks known as *time-driven cache attacks*, which uses execution time to glean secret information from the cipher. The ease with which time measurements can be made and the wide applicability of the attack makes the study of time-driven cache attacks important for present and future security systems.

In order to have a deeper understanding of the attack, the thesis categorizes time-driven cache attacks as either *profiled* or *non-profiled*, and develops analytical tools to evaluate the attack's success across implementations and platforms. The tools developed encapsulates and parametrizes block cipher algorithms, their implementations, cache memories for superscalar processors, and time-driven cache attack algorithms. These tools are used to analyze attack algorithms and to identify strategies that maximizes the attack's success. Further the tools are employed to pinpoint sources of leakage in block cipher algorithms and components in the cache memory, and thereafter quantify the information leakage due to each source. Specific to the leakage in cache memories, the thesis shows that micro-architectural acceleration features present in cache memories, such as pipelining, prefetching, non-blocking, and out-of-order servicing of cache misses can significantly affect information leakage. This has provided several counter-intuitive results such as cipher implementations with small look-up tables leaking more information than implementations that use large look-up tables, and that certain forms of time-driven cache attacks can be made more difficult by simply manipulating the size and number of look-up tables used in the implementation. This can result in implementations that are not only protected against time-driven cache attacks but also do not compromise on performance. The correctness of the tools developed have been extensively validated on Intel and AMD platforms and with simulators.