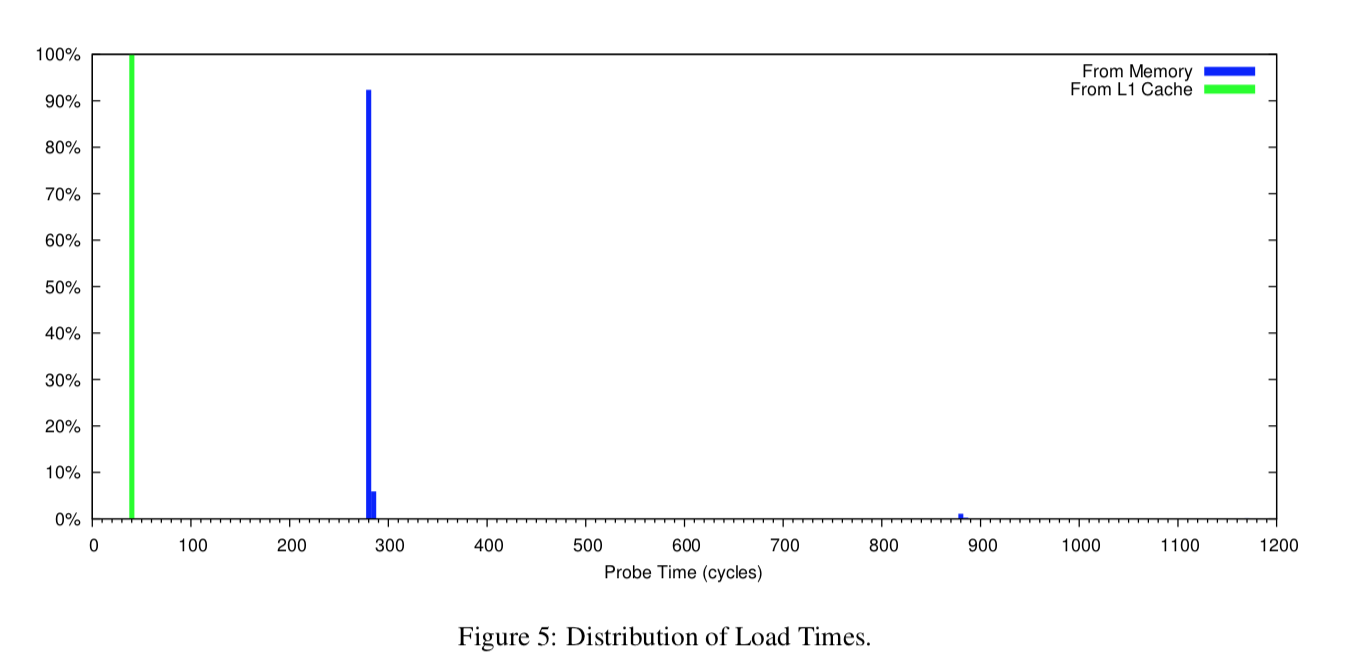
**ECE 396 Independent Study Report**

**Introduction**

This independent study focuses on cache side channel attack based on Prime + Probe attack and Flush + Reload attack. Side channel attack exploits the weakness of potential information leaks due to shared memory pages between non-trusting processes. Flush + Reload attacks and Prime + Probe attacks can be deployed across-core, across-VM and on multiple versions of GnuPG. Under an Infrastructure-as-a-service (IaaS) cloud-computing services, when a cryptographic library is being accessed by a victim process (or victim VM), a side-channel attack can occur to infer the message by an attacker process through shared caches and memory controllers. Once the attacker process has the confidential data, information can be communicated via a covert channel (covert channel attack) to a co-resident attacker VM.

**Background: Flush + Reload Attack**

Flush + Reload attack consists of three phases. At the first phase, the monitored memory cache line will be flushed by spy program. Then the spy program will wait for the victim to access those monitored cache lines at the second phase. At the last phase, the spy program will try to reload the cache line with contents from before during the first phase. If the access time for certain part of content is long, it means the corresponding cache line during the first phase has been flushed by the victim program. The corresponding cache line probably have victim program’s information. Base on access latency, a covert cache side channel could also be established to pass information from victim VM to attacker VM under Infrastructure-as-a-service (IaaS) cloud-computing services situation (Kayaalp). The graph below shows the access latency caused by accessing main memory compared with accessing cache.



(Kayaalp 6)

**Implementation of Single-thread Attack**

In order to have a better understanding of covert cache side channel attack, single-thread cache side channel attack and multithreading cache side channel attack have been implemented. Under single thread cache side channel attack, a sender and a receiver are in the same thread. Sender encodes its secret message by bringing certain information from parts of shared code into the cache, flush other parts of shared code. Receiver on the other end will try to access all the information from this shared code. When shared code in the cache is accessed, the access latency will be significantly shorter than those parts remained in main memory. Thus receiver can decode the message as either 1 or 0 based on data access latency.

**Encountered Problems:**

Memory Fence for rdtsc Measurement

The complier has optimization for executing the program. However, rdtsc for cycle measurement requires ordered execution of program to measure the correct time for accessing either main memory or cache. Thus memory fence is needed to prevent compiler from using optimized execution order.

Hardware Prefetching

Hardware has a tendency to prefetch instructions or contents from main memory to cache before the program even wants them. This hardware prefetching produces false information regarding to encoded message from sender side. Hardware perfecting can be prevented by randomizing the sequence to access the monitored cache lines.

Adjacency Prefetching

Hardware will prefetch adjacent cache lines for optimization. This will produce false information from the sender side too. The adjacency prefetching can be prevented by separating monitored cache lines apart (10 cache lines apart in the implementation).

**Implementation of Multithread Attack**

Under multithread cache side channel attack, two separate processes are running on separate cores. They are not able to directly share memory. By using mmap, two separated processes are able to share physical memory by sharing a common file. Sender will flush or not flush the virtualized address return back by mmap to encode 0 and 1. Receiver on the other end can measure the access latency based on the virtualized address from mmap.

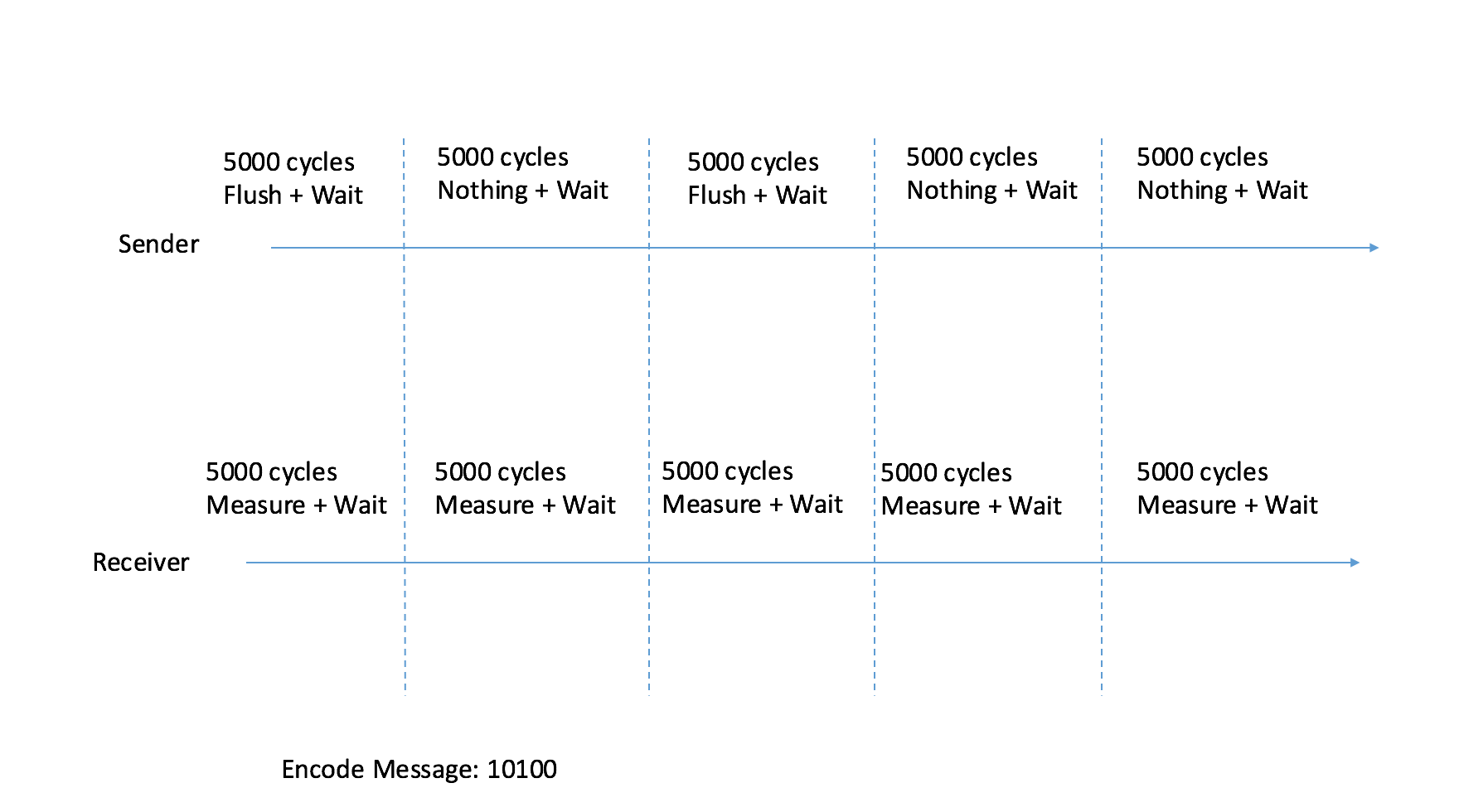
**Encountered Problems:**

Page Deduplication

mmap is used to make different processes across different processors share same physical memory page space. Virtualized memory addresses passed back by mmap can’t be shared by different processes, since different processes have different memory mappings. mmap can be used that a file can be treated as a sequence of memory spaces and the contents can be access by dereferencing pointers.

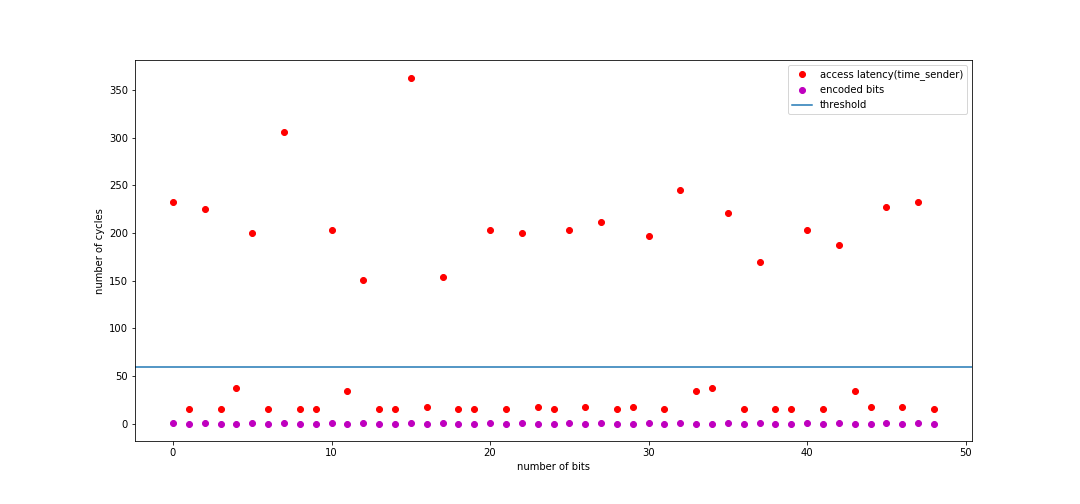
Synchronization Between Two Threads

In order to synchronize two threads, a common period of time for flush/nothing or measurement is established. Based on the measurement, the flush instruction is taking around 4500 cycles, thus the common period of time for sender and receiver is 5000 cycles. This scheme may cause variation in encoded message, because the sender and receiver might not be able to start to send or receive a bit at the same time. Delays between sender and receiver will cause problems. The scheme below is an ideal case.

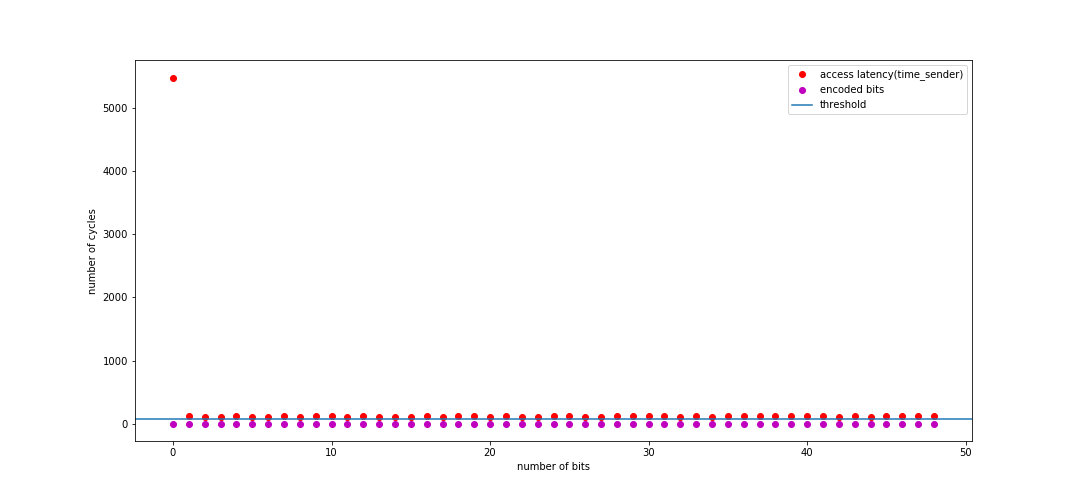


**Evaluation**

Under Single-thread Attack (Encoded Message: 10100):



Under Multithreading Attack (Encoded Message: 10100):



**Conclusion**

From this independent study, I have gained a clear understanding of two basic methods of cache side channel attack: Prime + Probe Attack and Flush + Reload Attack. By implementing Flush + Reload Attack under single thread and multithreading conditions, I have encountered some problem which led me to have a deeper understanding of hardware mechanisms such as hardware perfecting and adjacency perfecting. I hope that I could be able to conduct formal research in hardware architecture design and hardware security field as I gain more knowledge from more advanced level architecture courses.

**References**

Kayaalp, Mehmet, et al. “A High-Resolution Side-Channel Attack on Last-Level Cache.” *Proceedings of the 53rd Annual Design Automation Conference on - DAC '16*, 2016, doi:10.1145/2897937.2897962.