Flush-Based Attacks

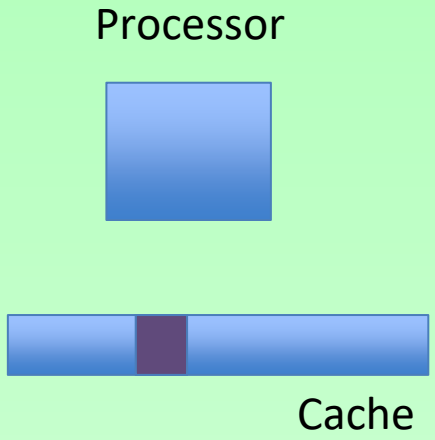
Today

- Flush+Reload
- Recovering missing bits from RSA keys
- Performance degradation attacks
- Flush+Flush

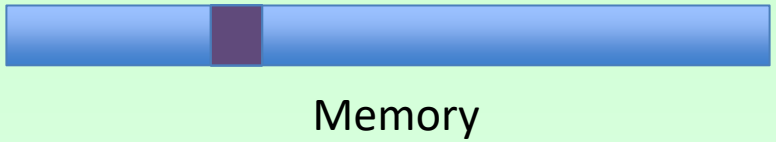
Bridging the gap

Cache utilises locality to bridge the gap

- Divides memory into *lines*
- Stores recently used lines

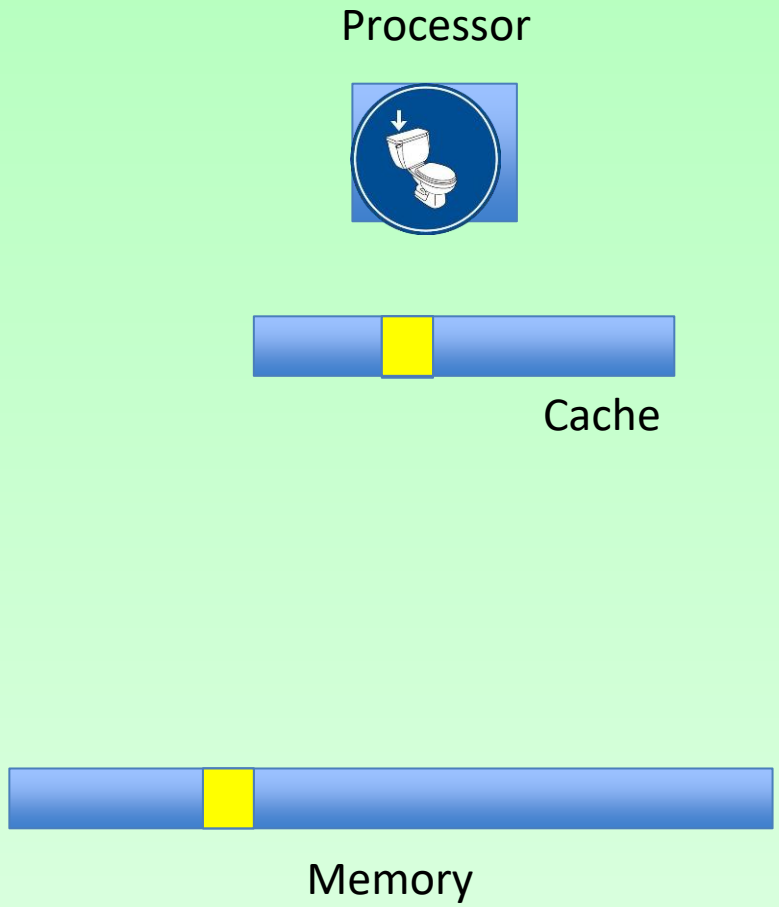


- In a *cache hit*, data is retrieved from the cache
- In a *cache miss*, data is retrieved from memory and inserted to the cache



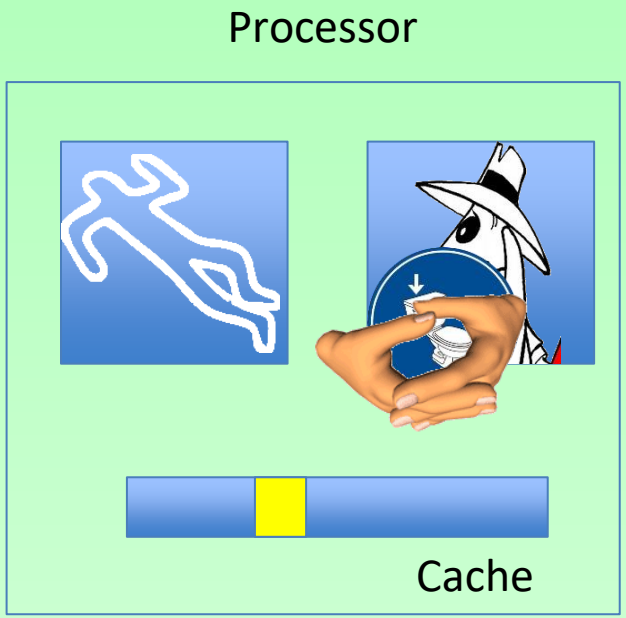
Cache Consistency

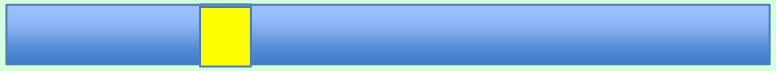
- Memory and cache can be in inconsistent states
  - Rare, but possible
- Solution: Flushing the cache contents
  - Ensures that the next load is served from the memory



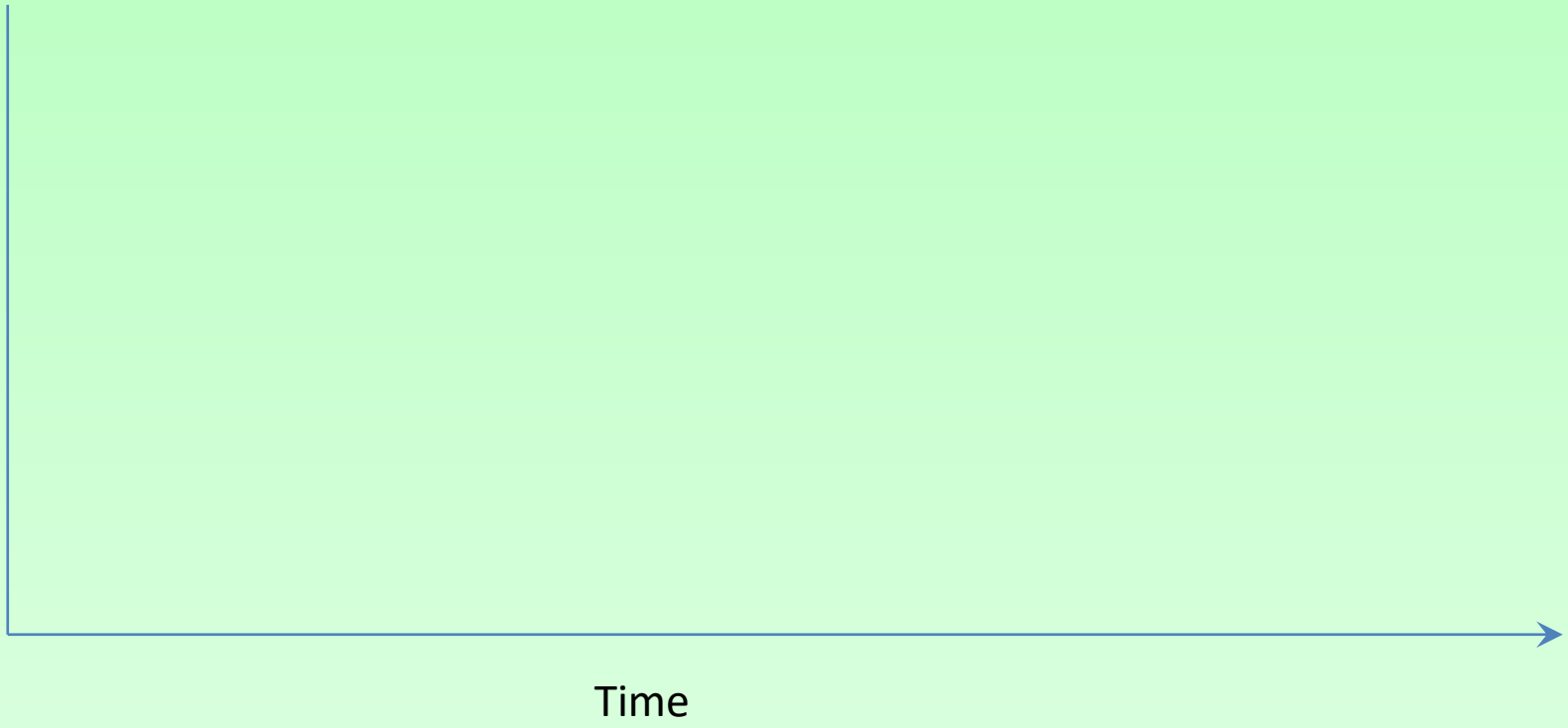
The FLUSH+RELOAD Attack

- Flush a memory line from the cache
- Wait a bit
- Measure time to load line
  - slow-> no access
  - fast-> access
- Repeat

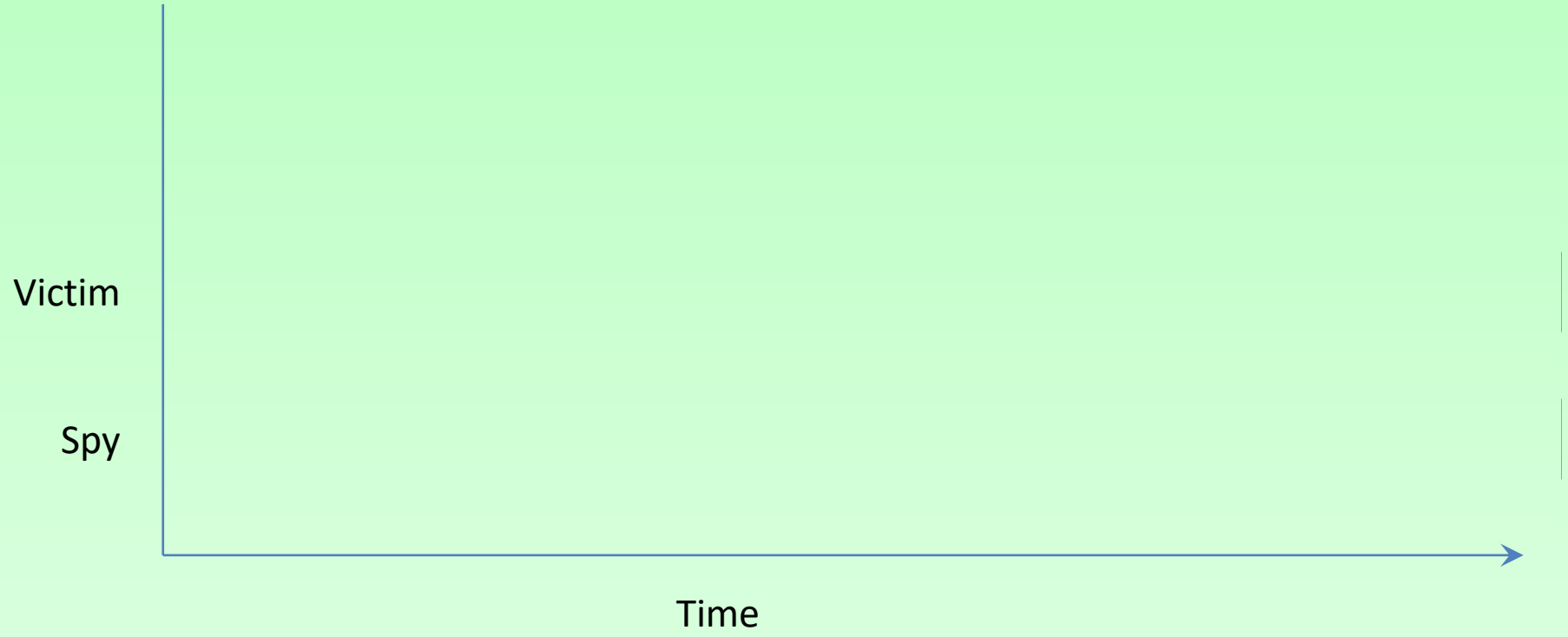




Flush+Reload

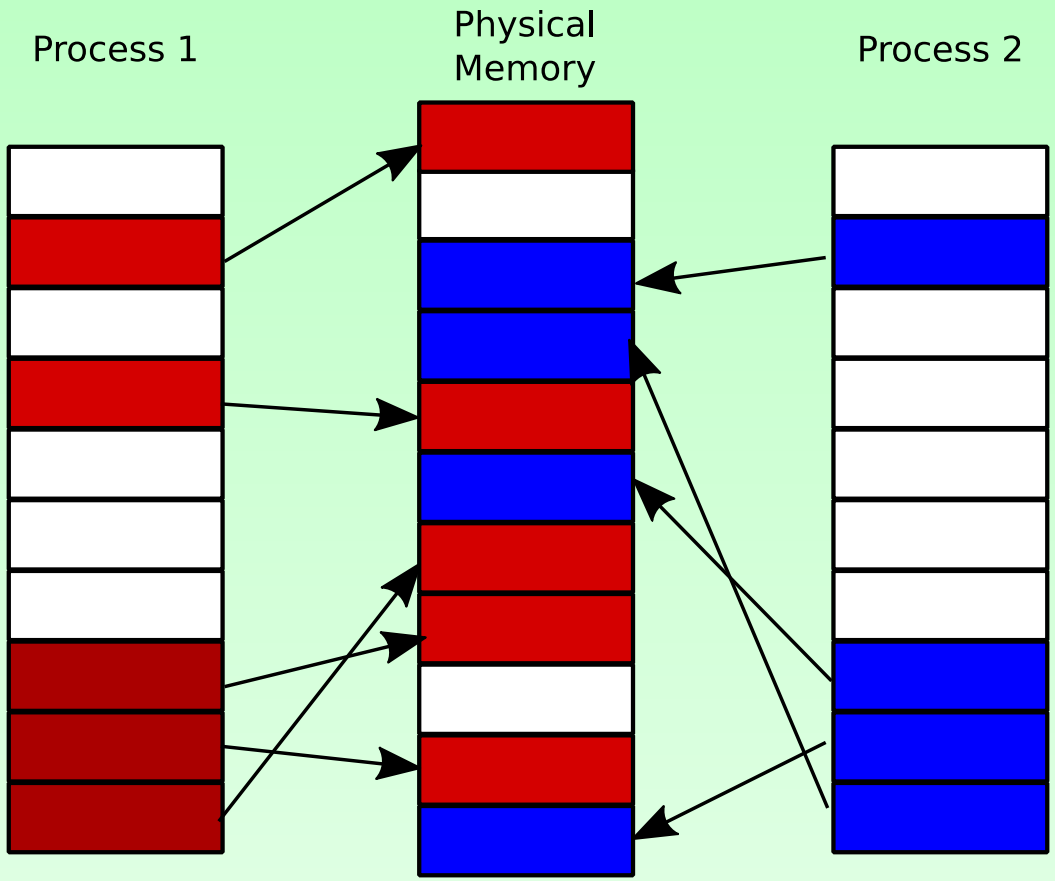


Flush+Reload



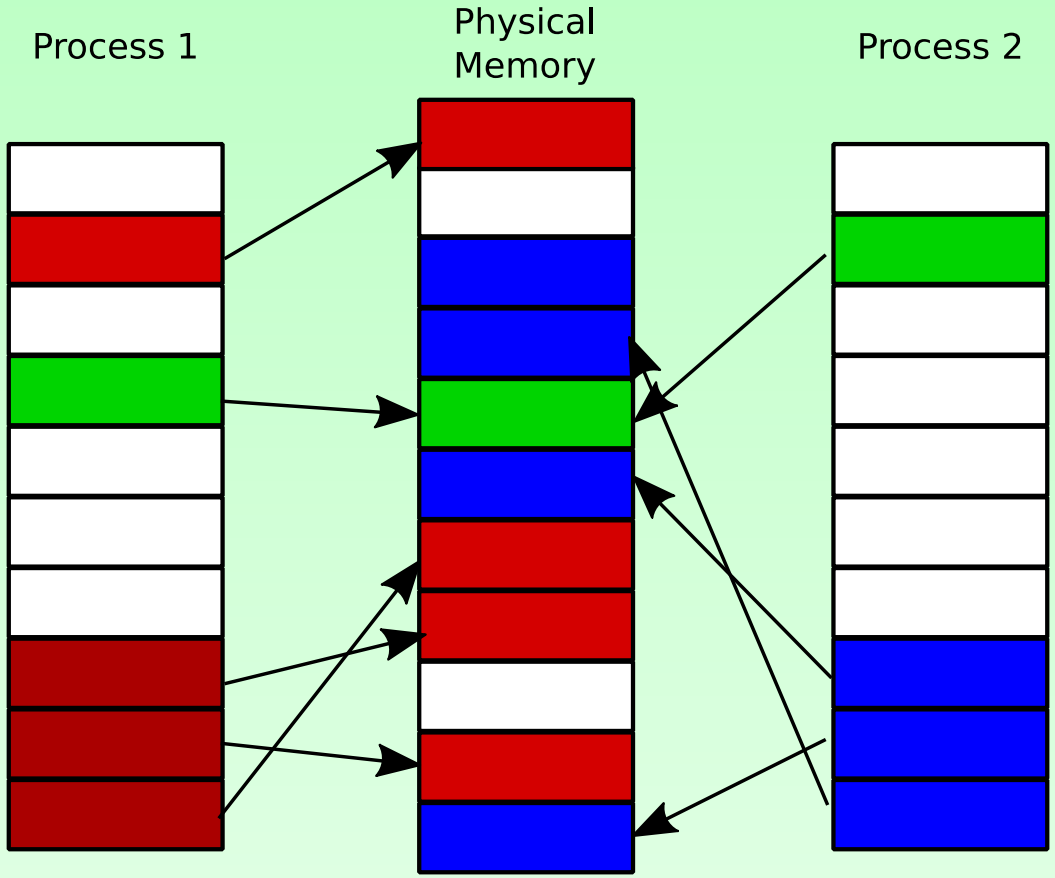
Detour - Virtual Memory

- Processes execute within a virtual address space
  - Virtual pages map to physical frames



Sharing

- Frames can be shared by multiple processes
  - Read only sharing maintains *functional* isolation
  - Protection using *Copy-on-write*



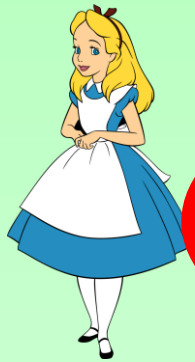
Causes of sharing

- Content-aware sharing
  - Pages from the same file have identical content
  - Shared program or library code
    - Can also share constant data
  - Shared images in PaaS clouds
- Content-based sharing (a.k.a. page deduplication)
  - The system identifies and coalesces identical pages
  - Implemented in many hypervisors and in most modern operating systems

Demo

The RSA Encryption System

- The RSA encryption is a public key cryptographic scheme



$M = C^d \bmod N$

M

$C = M^e \bmod N$

Key Generation:

- Select random primes p and q
- Calculate $N = pq$
- Select a public exponent e(=65537)
- Compute $d=e^{-1} \bmod \varphi(N)$
- (N, e) is the public key
- (p, q, d) is the private key

Square and Multiply Exponentiation

```
x ⟵1
for i ⟵|d|-1 downto 0 do
    x ⟵x^2 mod n
    if (d_i = 1) then
        x = xC mod n
    endif
done
return x
```

Operation	x	i	d_i

The private key is encoded in the sequence of operations !!!

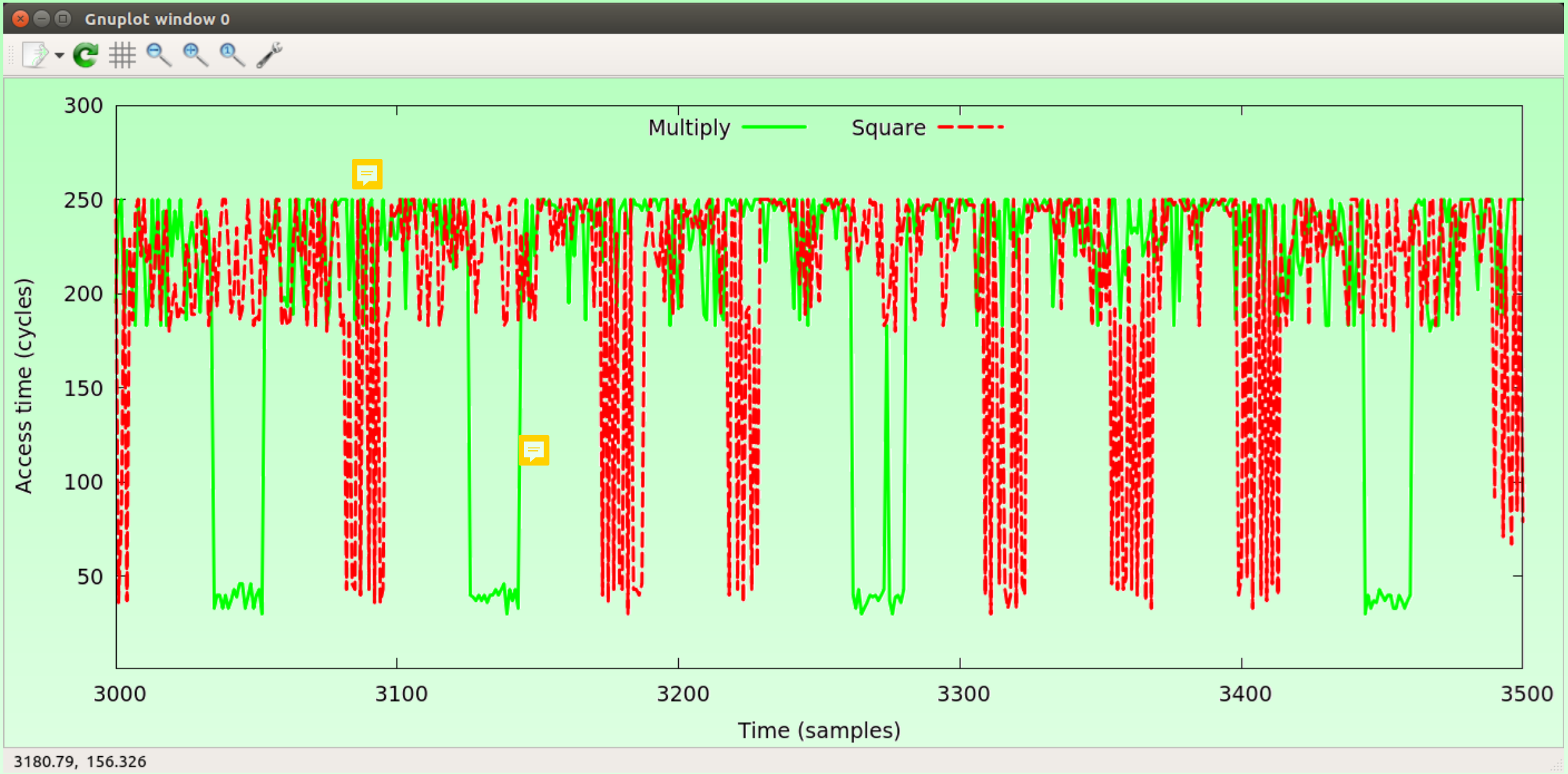
Example:

$11^5 \bmod 100 =$

$161{,}051 \bmod 100 = 51$

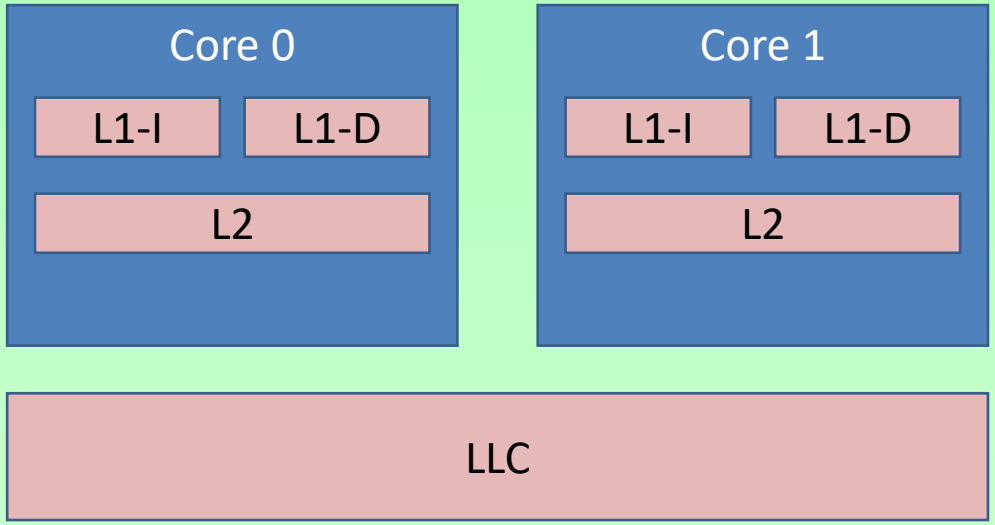
Demo

Flush+Reload on Square-and-Multiply



Why not AMD?

- Multilevel cache hierarchy



- Inclusive LLC: stores copies of all data in higher level caches
- Exclusive LLC: does not store data cached in higher level caches
- Non-inclusive LLC: not strictly inclusive. (Can be exclusive.)

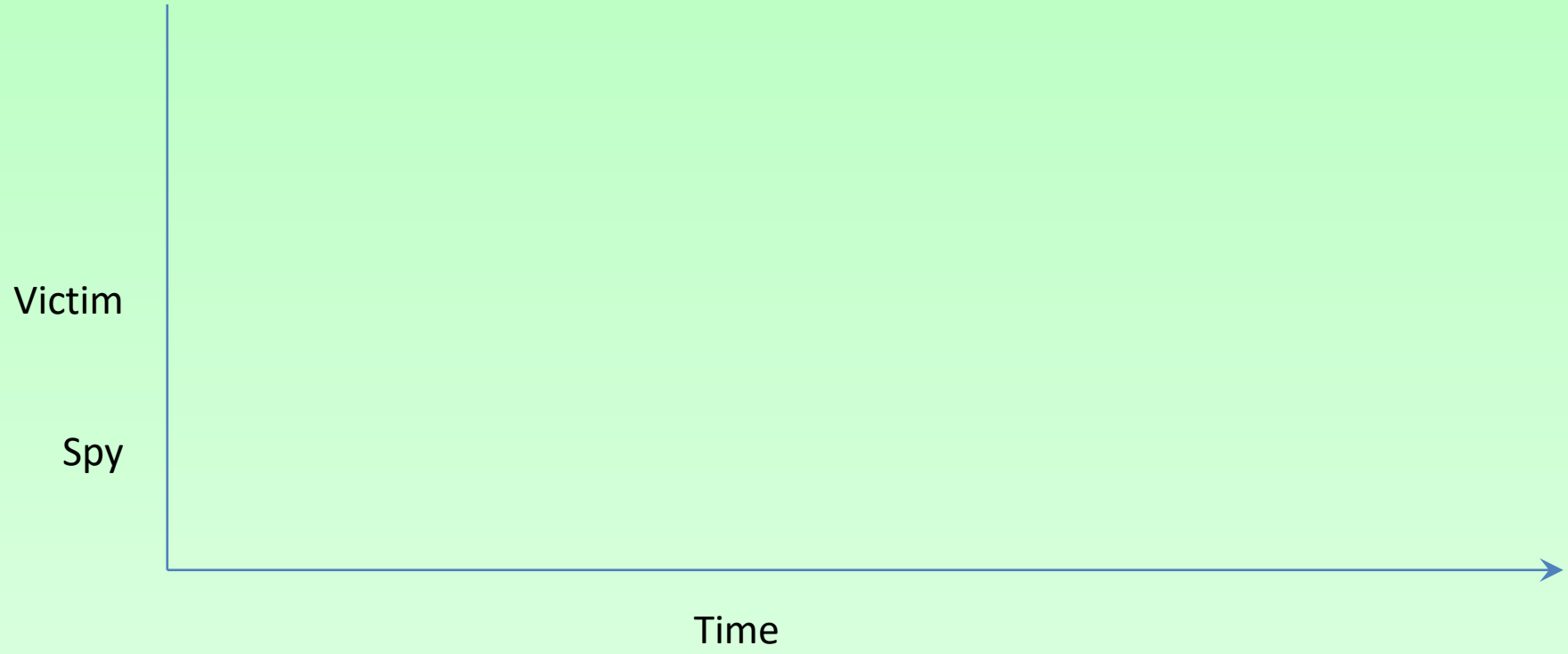
Defences

Performance Degradation

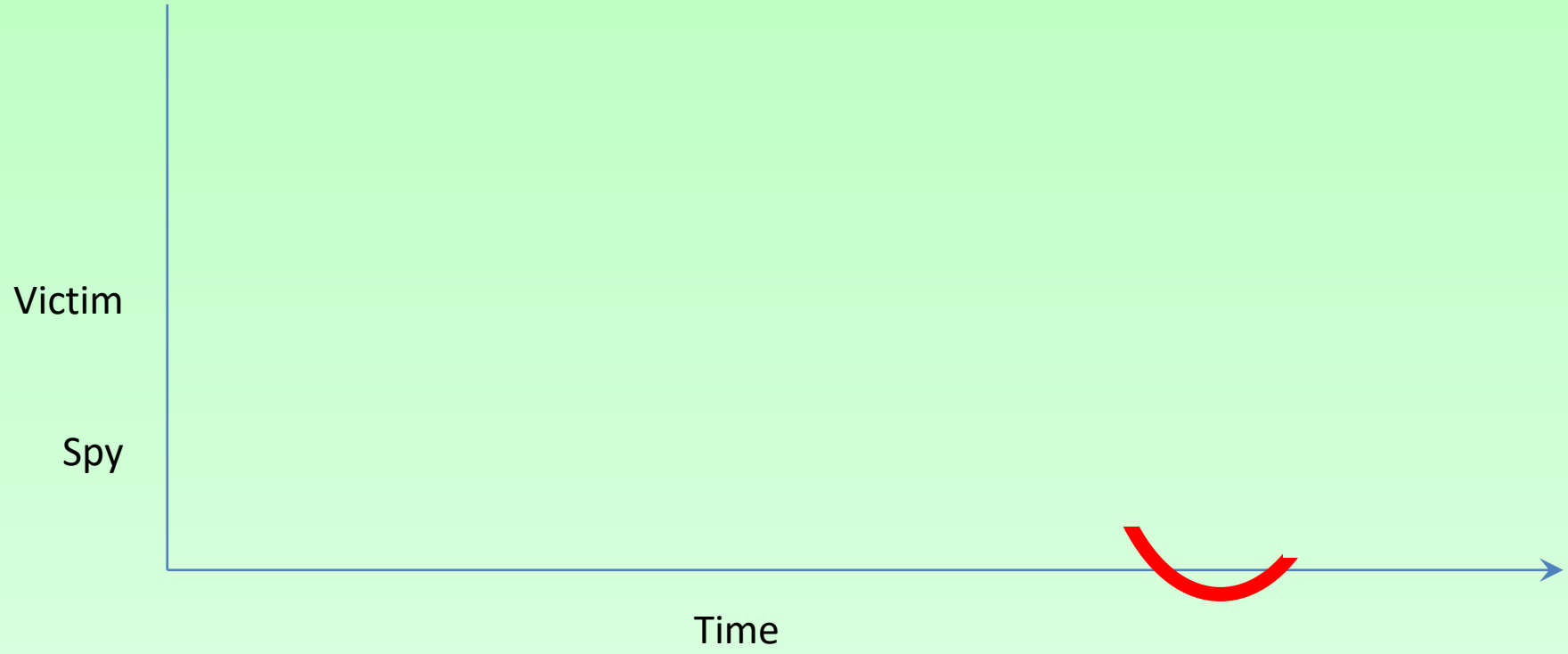
Allan et al. “Amplifying Side Channel Through Performance Degradation” ACSAC 2016

See also: Cabrera Aldaya and Brumley. “HyperDegrade: From GHz to MHz Effective CPU Frequencies”, USENIX Security 2022

Flush+Reload



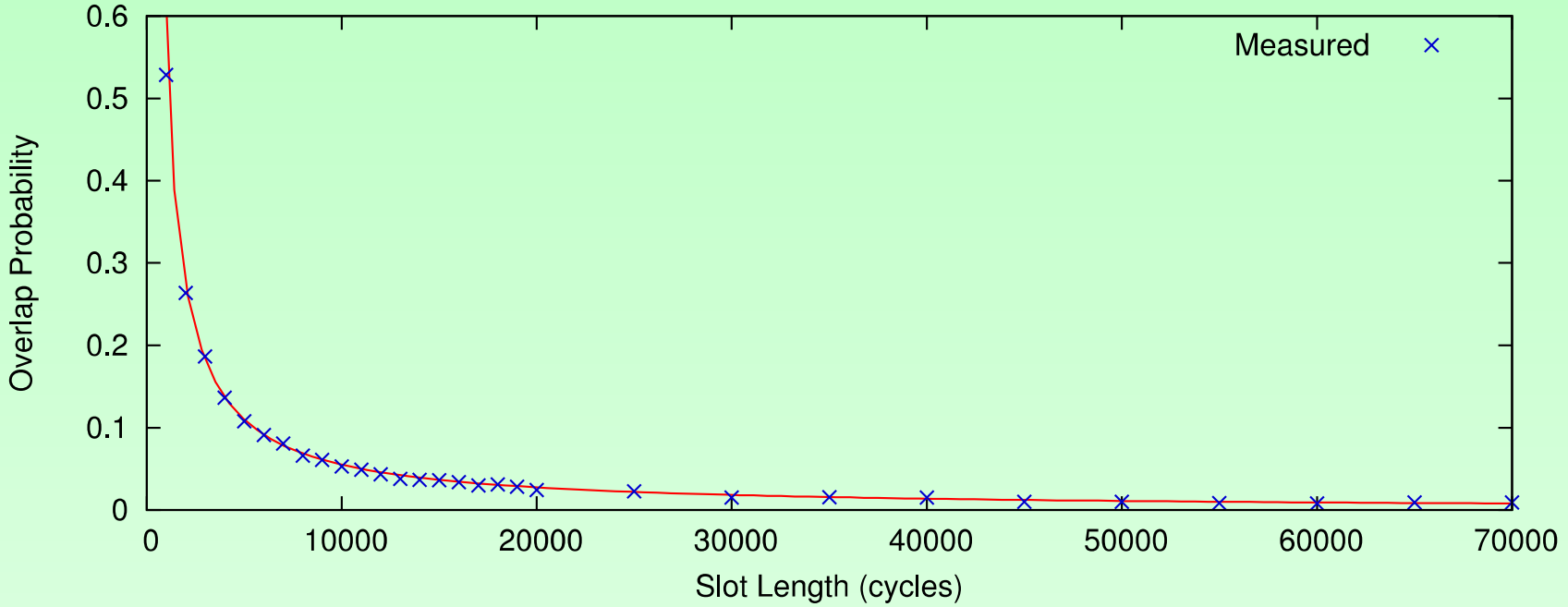
Timing matters



Access missed due to temporal overlap

Probability of a probe miss

- Ratio of probe time to slot length



Probing OpenSSL ECDSA

- For secp256k1, operation times are:
  - Addition: 2,892 cycles
  - Doubling: 3,086 cycles
  - Everything else: negligible

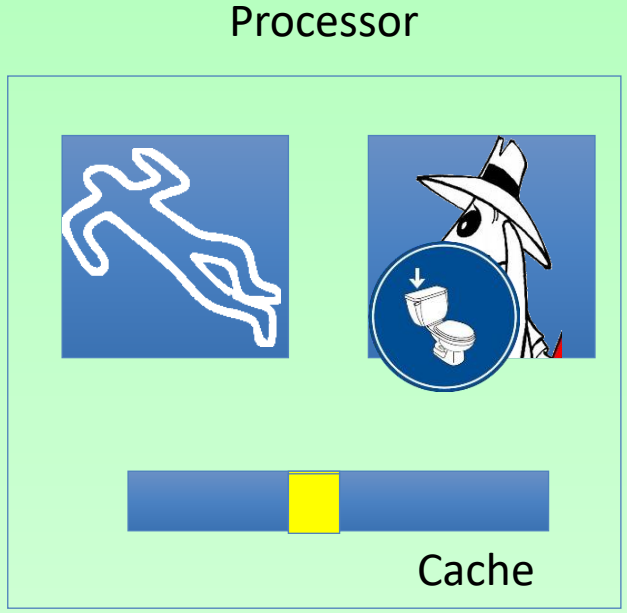
- To avoid aliasing errors, need to probe at least once every 1,450 cycles

- Limitations on probing
  - Each probe takes 450 cycles.
  - At the required probe rate, we can expect a 37% miss rate

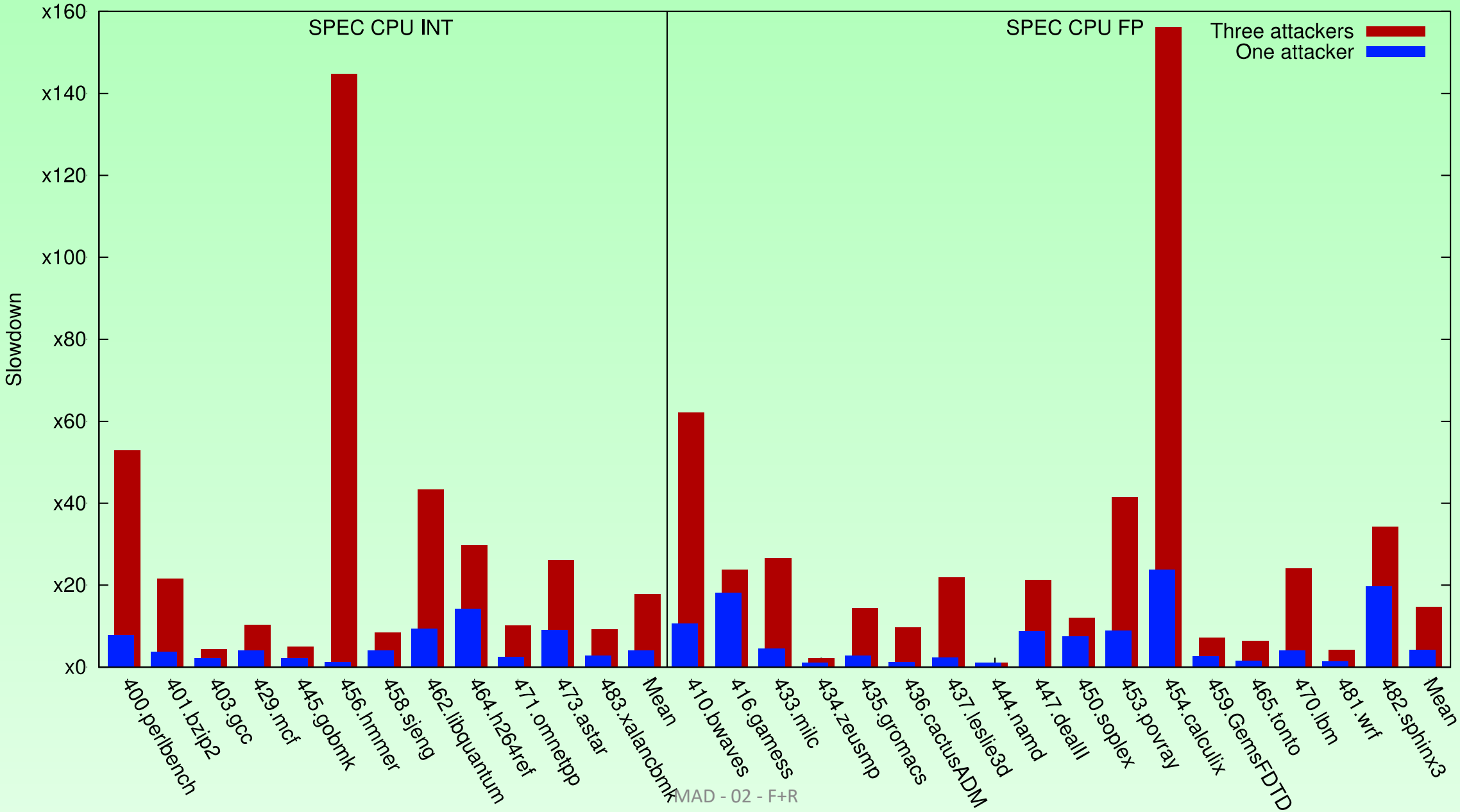
Solution: slow the victim down

- Flushing data or code forces the victim to retrieve it from the memory

- Performance degradation attack:
  - Find frequently used code lines
    - For example, usingprofiling information
  - Repeatedly flush them from the cache



SPEC Results



Attacking OpenSSL

- Use two threads to flush three addresses spanning the inner loop in the bignum multiplication code
  - Add slowed by a factor of 53 (153,709 cycles)
  - Double slowed by a factor of 41 (126,282 cycles)

- Monitor four memory locations: one in add, one in double and two in the call chain of invert

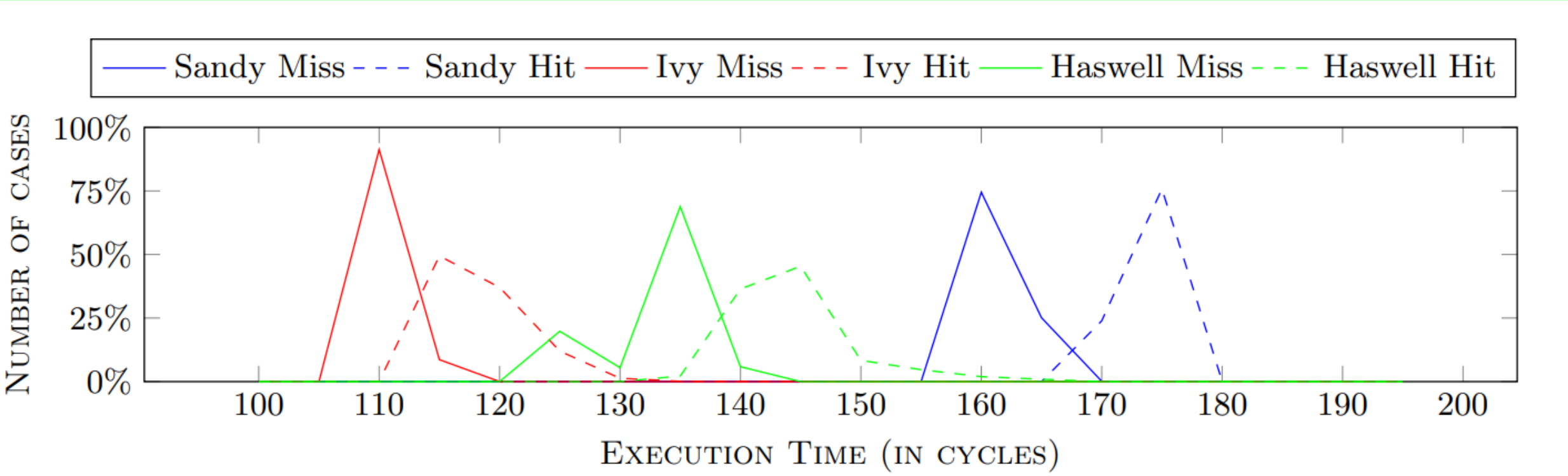
- Probe each monitored location once every 17,000 cycles

Flush+Flush

Gruss et al. “Flush+Flush: A Fast and Stealthy Cache Attack”, DIMVA 2016

Flush+Flush

- Flush time depends on cache residency



- Small differences, variable baseline

Improvements

- Timing depends on core and slice. Can improve accuracy through calibration
  - Didier and Maurice, “Calibration Done Right: Noiseless Flush+Flush Attacks.”, DIMVA 2021

- Avoiding fences increases difference between hit and miss
  - Ding et al., “A Cross-Platform Cache Timing Attack Framework via Deep Learning”, DATE 2022

Summary

- Flush+Reload and related attacks
- Amplification through performance degradation

- Next: Prime+Probe Attacks
  - Read: Osvik, Shamir, and Tromer, “Cache attacks and Countermeasures: The Case of AES”, CT-RSA 2006