PMU-Leaker: Performance Monitor Unit-based Realization of Cache Side-Channel Attacks

Pengfei Qiu[1,2], Qiang Gao[1], Dongsheng Wang[2], Yongqiang Lyu[2], Chunlu Wang[1], Chang Liu[2], Rihui Sun[3], Gang Qu[4]

[1]Beijing University of Posts and Telecommunications

[2]Tsinghua University [3]Harbin Institute of Technology

[4]University of Maryland, College Park

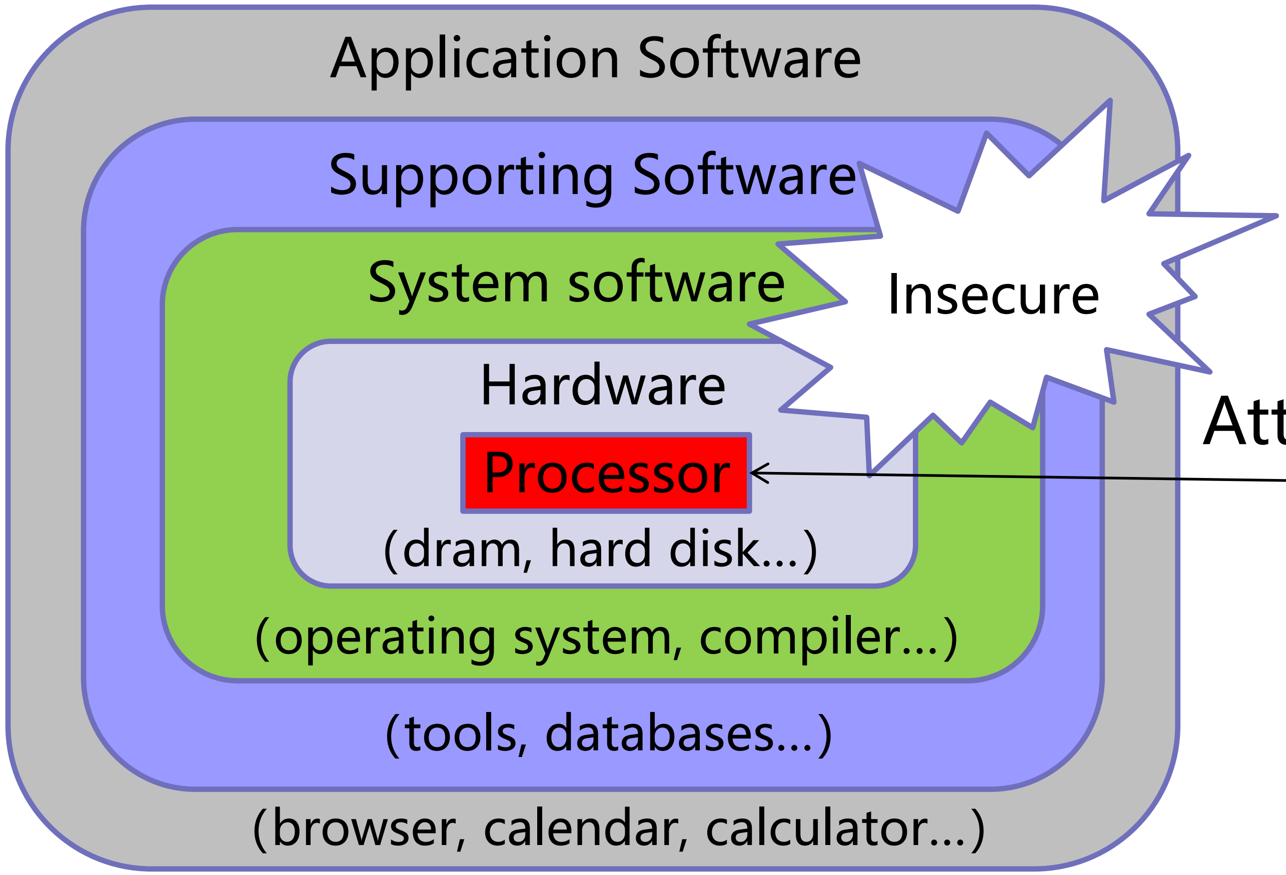
gangqu@umd.edu

Self-Introduction

- Basic information
  - 2015: B.S. degree from Harbin Institute of Technology
  - 2020: Ph.D. degree from Tsinghua University
  - 2022: Post doctor at Tsinghua University
  - Now: Associate professor at Beijing University of Posts and Telecommunications

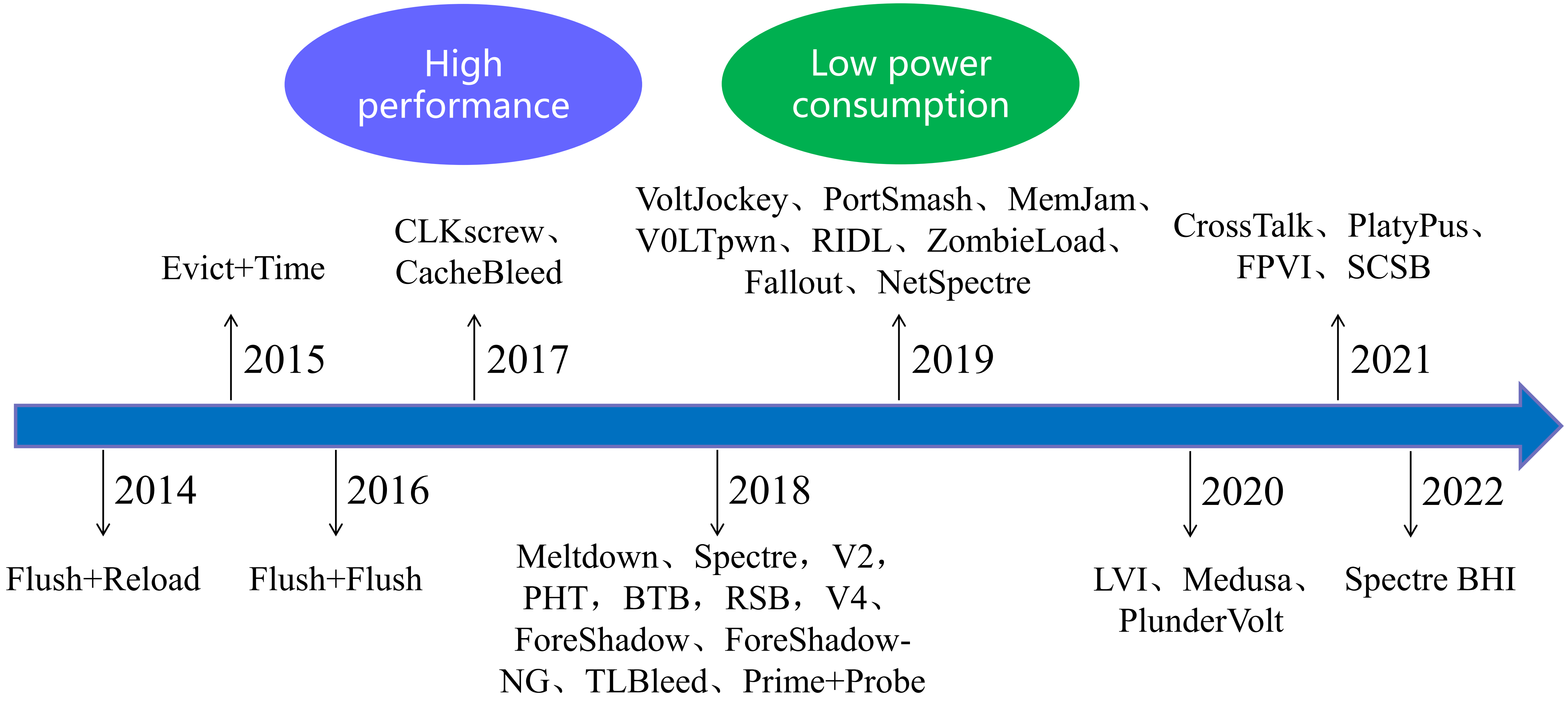
- Research interest——Hardware security
  - Hardware vulnerability mining
  - Secure architecture design.

The Keystone of the Computer Security



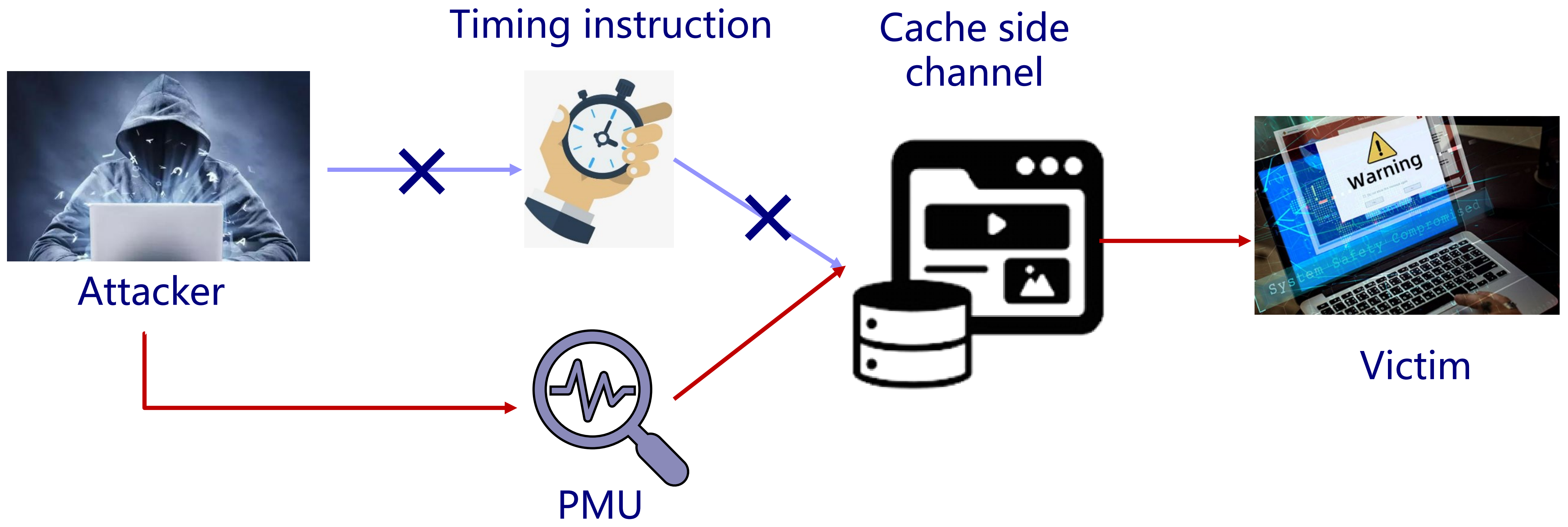
Processor Vulnerabilities

- Main optimization objectives in traditional processor design


High performance
Low power consumption
Evict+Time
2015
CLKscrew、CacheBleed
2017
VoltJockey、PortSmash、MemJam、V0LTpwn、RIDL、ZombieLoad、Fallout、NetSpectre
2019
CrossTalk、PlatyPus、FPVI、SCSB
2021
2014
Flush+Reload
2016
Flush+Flush
2018
Meltdown、Spectre，V2，PHT，BTB，RSB，V4、ForeShadow、ForeShadow-NG、TLBleed、Prime+Probe
2020
LVI、Medusa、PlunderVolt
2022
Spectre BHI

Our Work

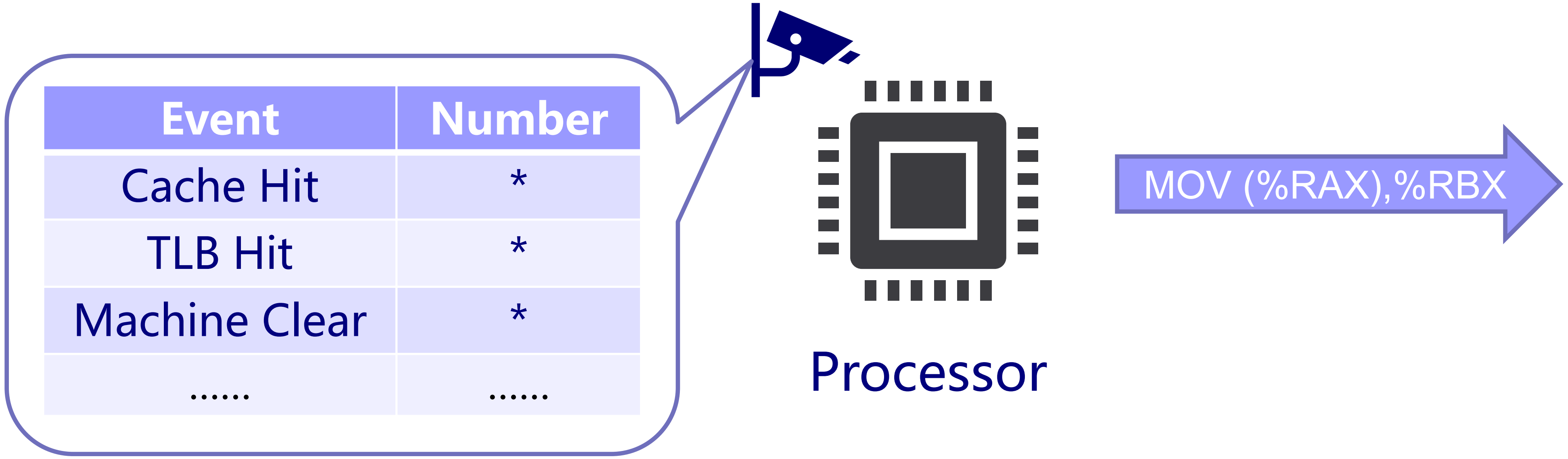
- Focus on the security of the processor`s performance monitor unit
- PMU-Leaker
  - Enable cache side channel attacks even when the timing instruction is unavailable
  - PMU identifies whether a data-load operation hits cache or misses cache



Outline

Performance Monitor Unit (PMU)

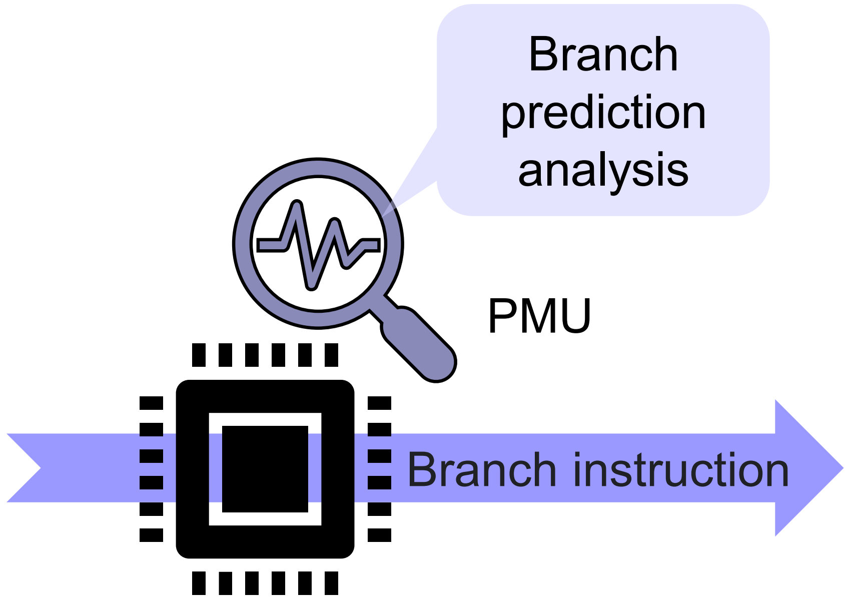
- A hardware module
  - Measure various architectural and microarchitectural events
    - Instructions that are assigned to a port
    - Cache misses/hits at different levels
    - Mispredicted branches

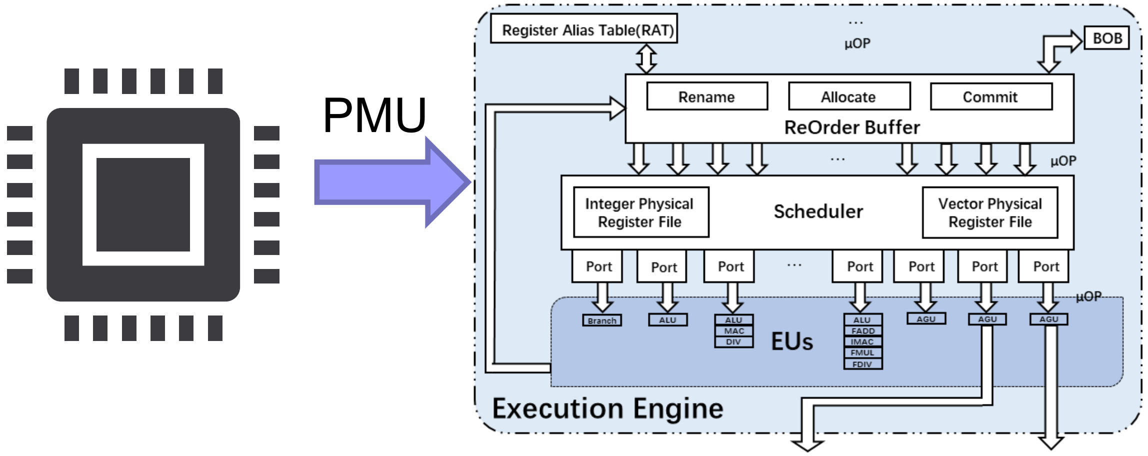


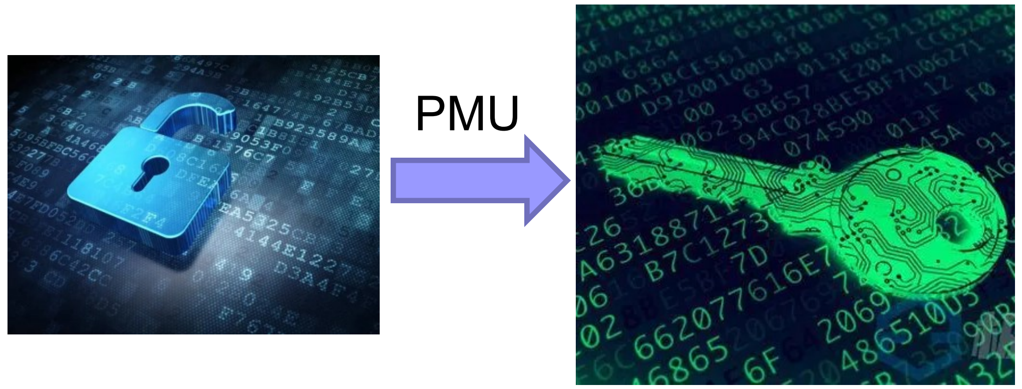
Event	Number
Cache Hit	*
TLB Hit	*
Machine Clear	*
……	……

Utilizations of PMU

- Function
  - Performance analysis and optimization
  - Reverse engineer black-box processors
  - Infer keys of encryption algorithms

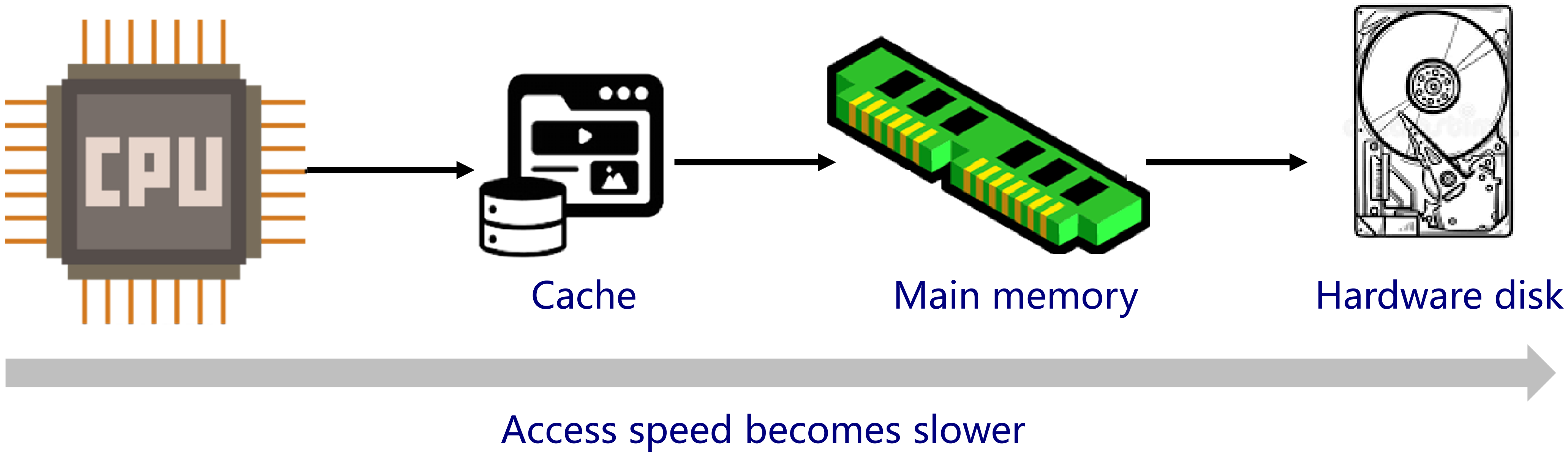






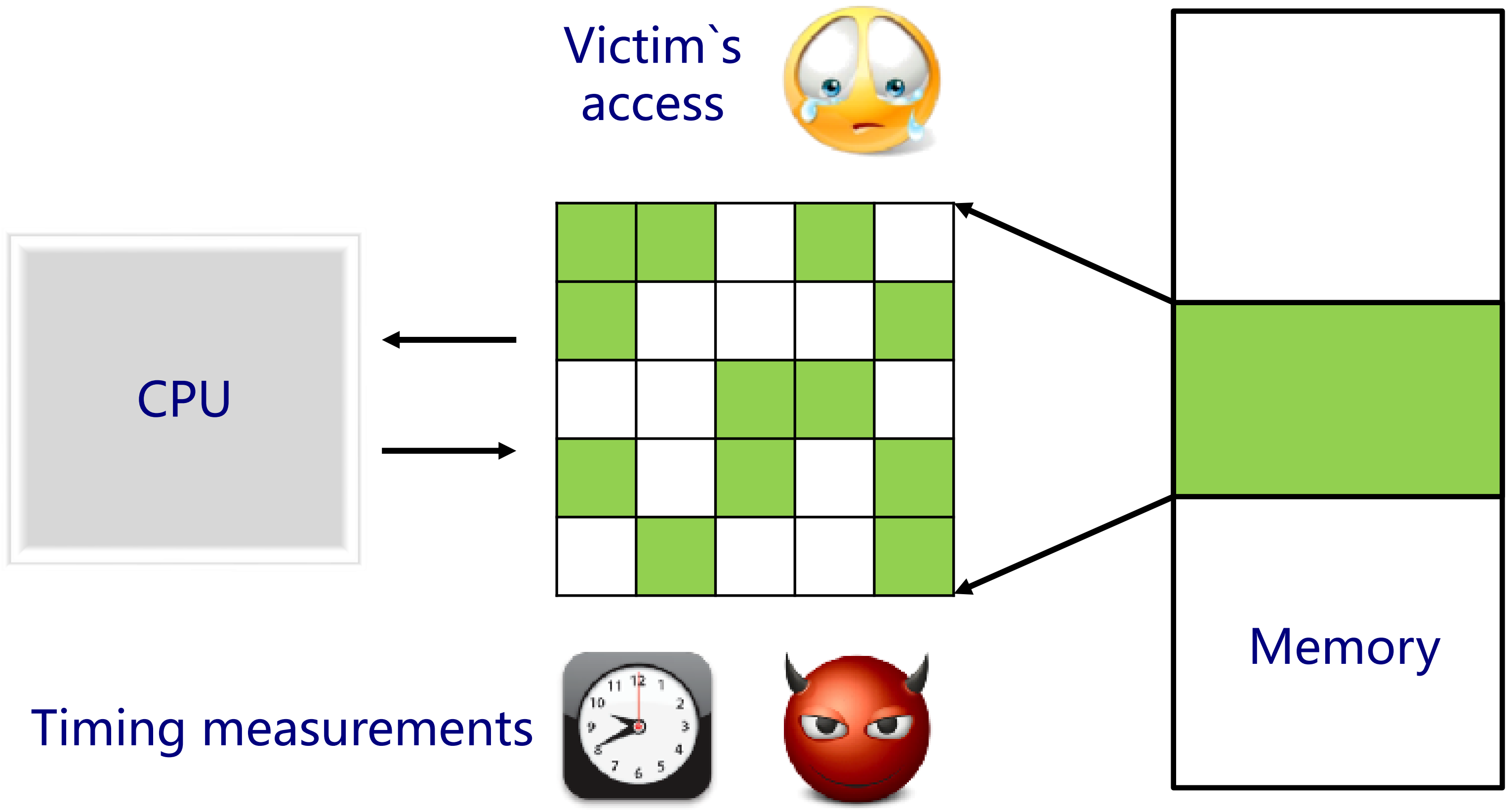
Cache

- A high-speed storage media
  - Between CPU cores and memory
  - Reduce the time to access data from the main memory
- Feature
  - Data access time when missing cache is higher than hitting cache



Cache Side-Channel Attack

- Based on the time difference between cache hits and misses
  - ➢Infer keys of encryption algorithms (e.g. AES, RSA, and ECC)
  - ➢Implement the transient execution attacks to leak secret data



Prime+Probe

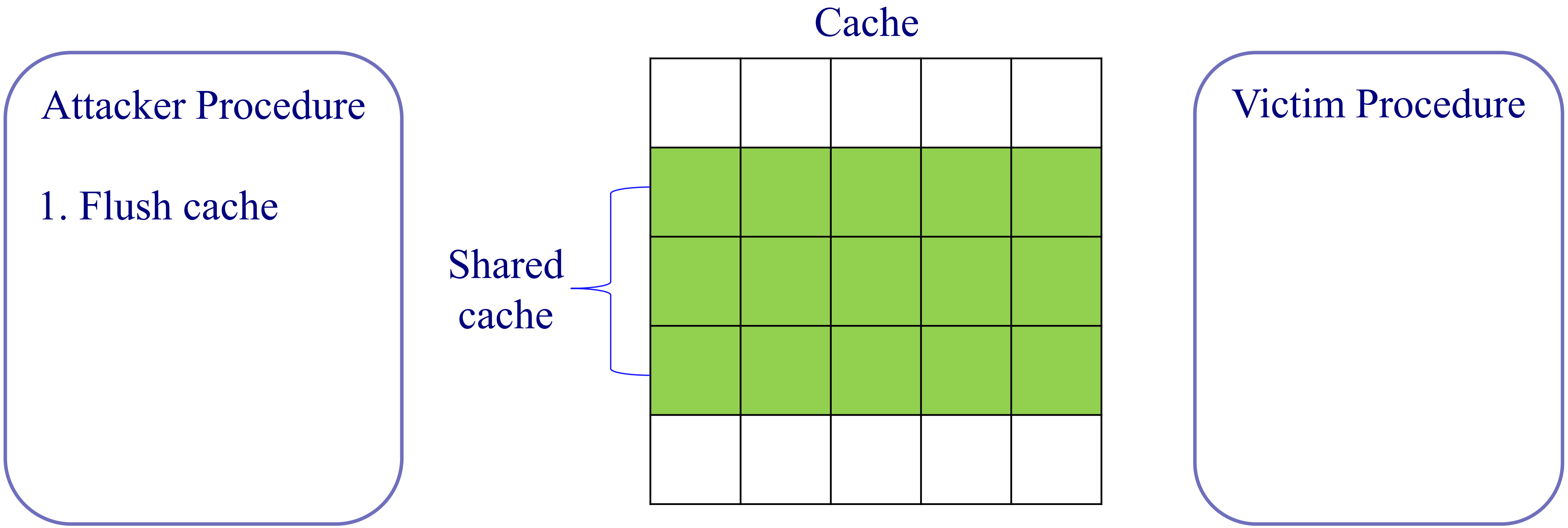
Evict+Time

Flush+Flush

Flush+Reload

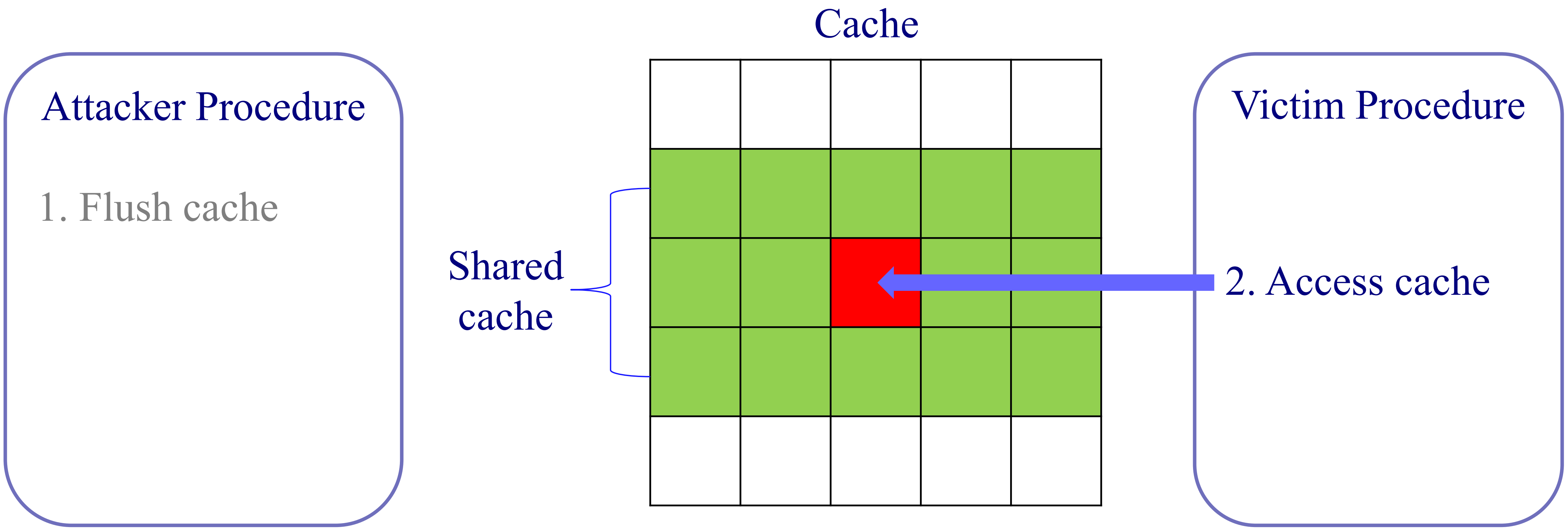
Example of Cache Side-Channel Attack——Flush+Reload

- The first step
  - Attacker procedure flushes data of the shared cache



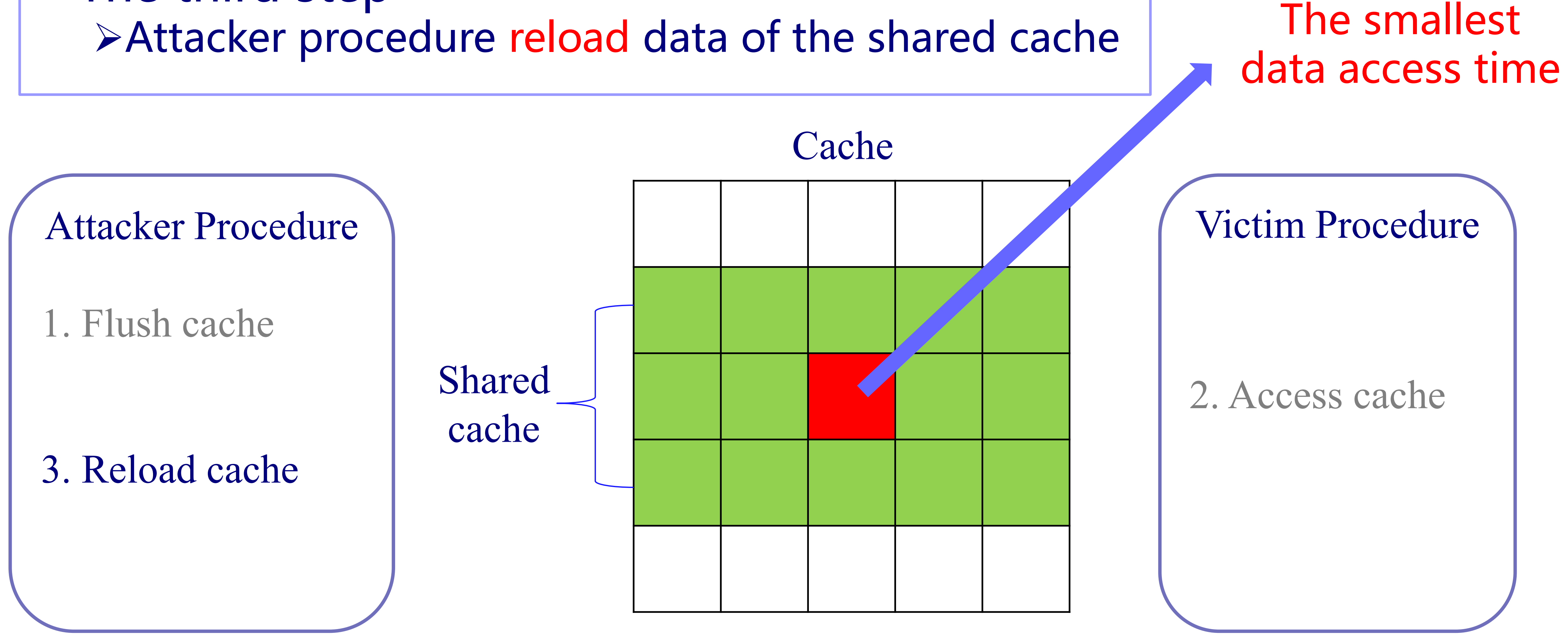
Example of Cache Side-Channel Attack——Flush+Reload

- The second step
  - ➢Victim procedure access data of the shared cache



Example of Cache Side-Channel Attack——Flush+Reload

- The third step
  - ➢Attacker procedure reload data of the shared cache



Our Contributions

- The current cache side channel attacks
  - Rely on the accurate timing instructions (e.g. rdtsc)
  - Can be simply mitigated by disabling the accurate time measurements

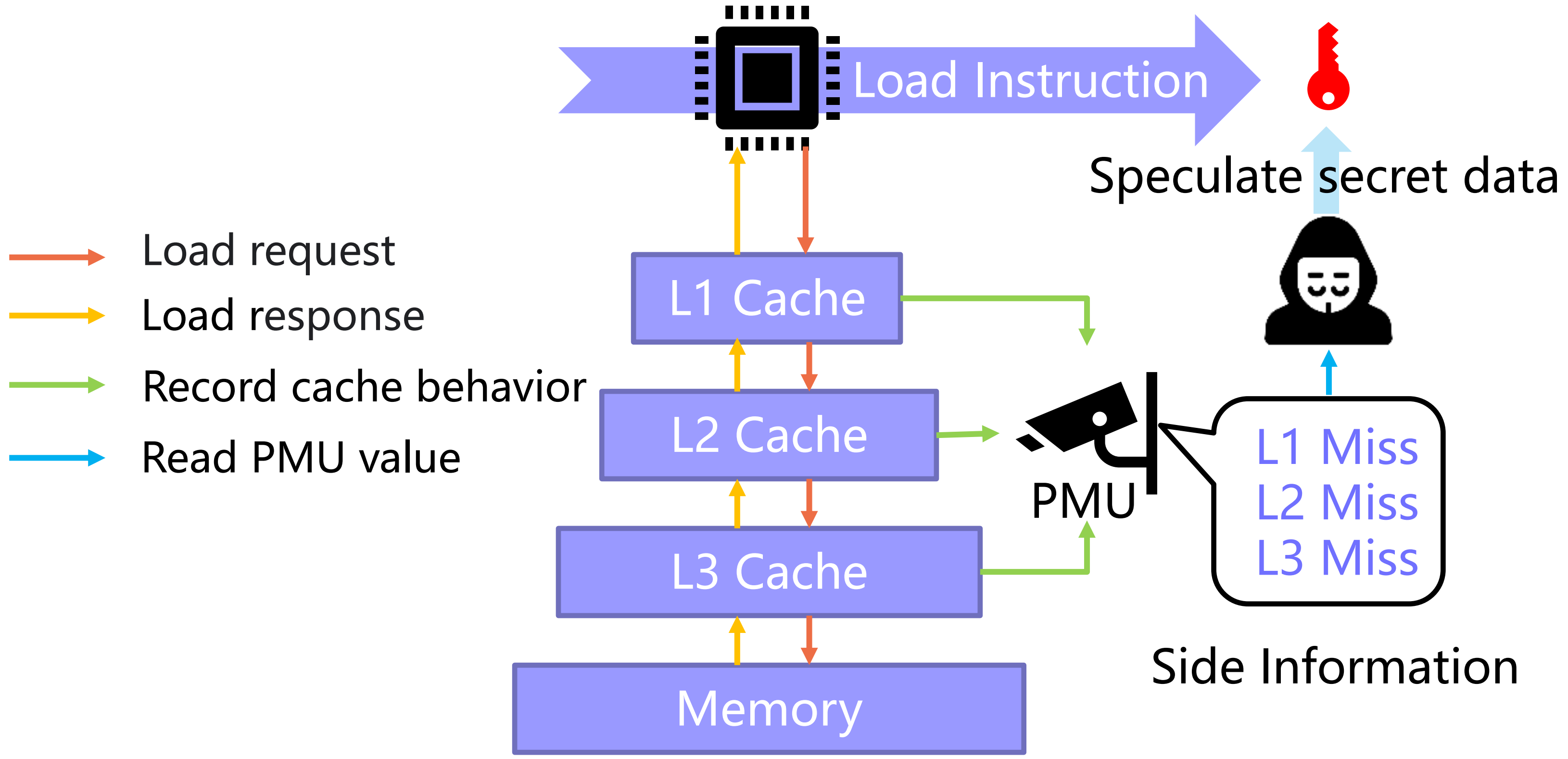
Our contributions

- Discover that PMU counters can reveal cache hits/misses
- PMU-Leaker: replace time measurements to implement cache side-channel attacks
- 24 PMU events to realize Flush+Realod cache side channel in transient execution attacks
- 6 PMU events to realize the Prime+Probe cache side channel in speculating the key of a victim AES performed in SGX
- CPU_CLK_THREAD_UNHALTED.REF_XCLK can measure instruction execution time

Outline

Motivation

- The events monitored by PMU expose interactions between instructions and hardware resources



Can the information leaked from PMU replace the accurate time measurements for cache side-channel attacks?

Assumption and Threat Model

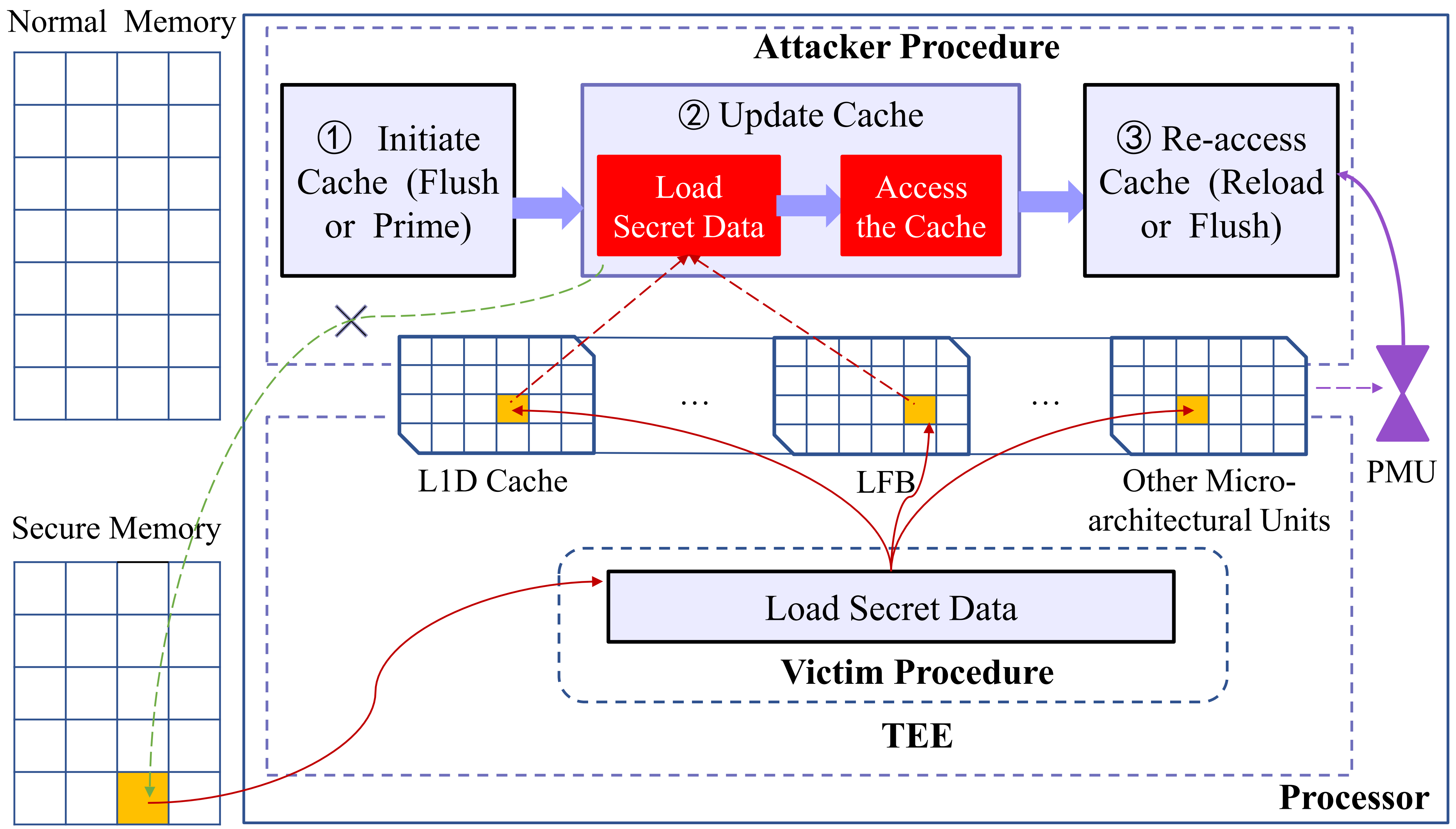
Assumption

- PMU is enabled in the target processor
  - Most processors use PMU as a performance profiling technology
- The attackers have a root privilege

Threat Model

- Attackers use PMU to implement cache side channel attack
  - Infer keys of encryption algorithms performed in TEE
  - Recover secret data of TEE with transient execution attacks

Overview of PMU-Leaker

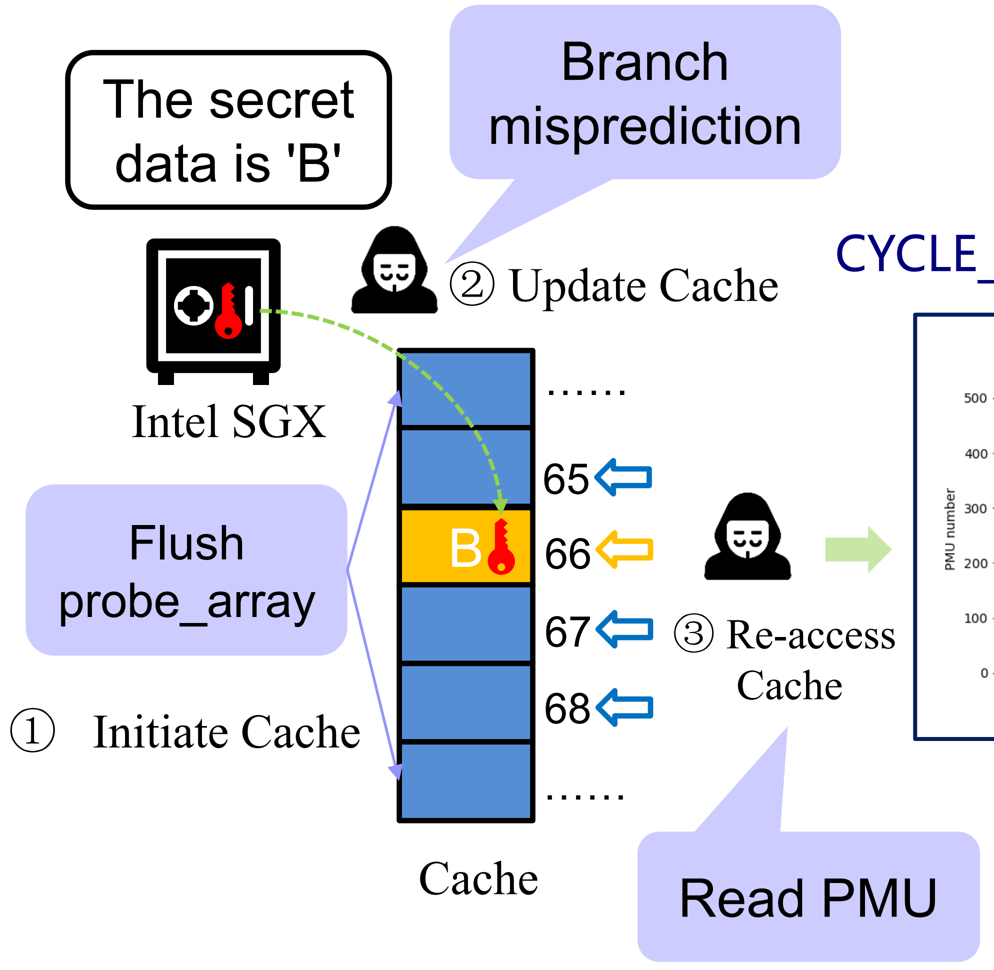


Attack Target

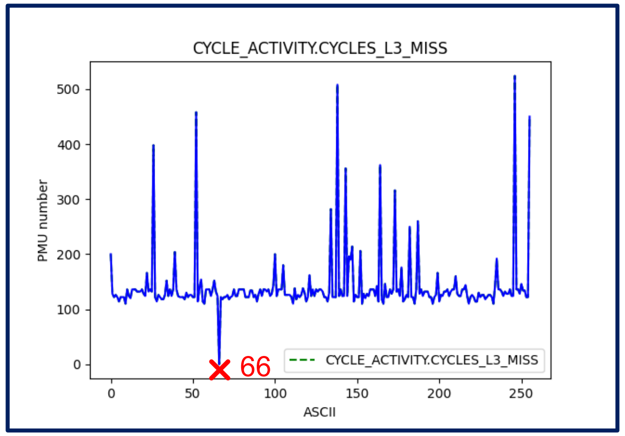
Keys of encryption algorithms/secret data stored in TEE

- The root privilege is out of the Trusted Computing Base of TEE
- SGX is vulnerable to the cache side channel attacks
- Several transient execution attacks have been verified feasible to leak data from SGX including Foreshadow, ZombieLoad, and Spectre

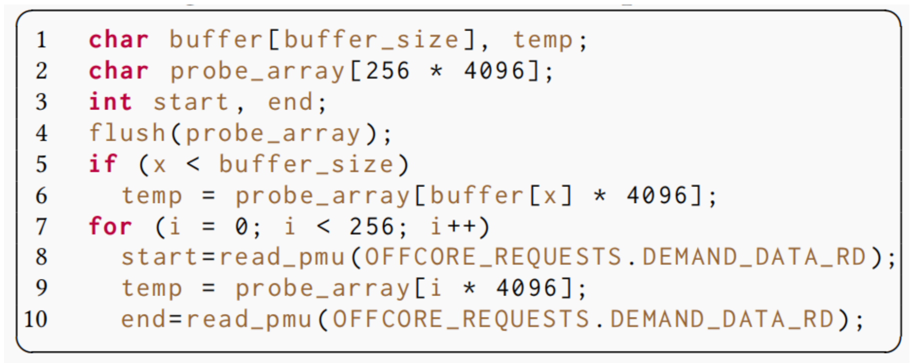
Case Study: Spectre Attack based on PMU-Leaker



CYCLE_ACTIVITY.CYCLES_L3_MISS



Example



```
char buffer[buffer_size], temp;
char probe_array[256 * 4096];
int start, end;
flush(probe_array);
if (x < buffer_size)
  temp = probe_array[buffer[x] * 4096];
for (i = 0; i < 256; i++)
  start=read_pmu(OFFCORE_REQUESTS.DEMAND_DATA_RD);
  temp = probe_array[i * 4096];
  end=read_pmu(OFFCORE_REQUESTS.DEMAND_DATA_RD);
```

Gadget

Outline

Experiment setup

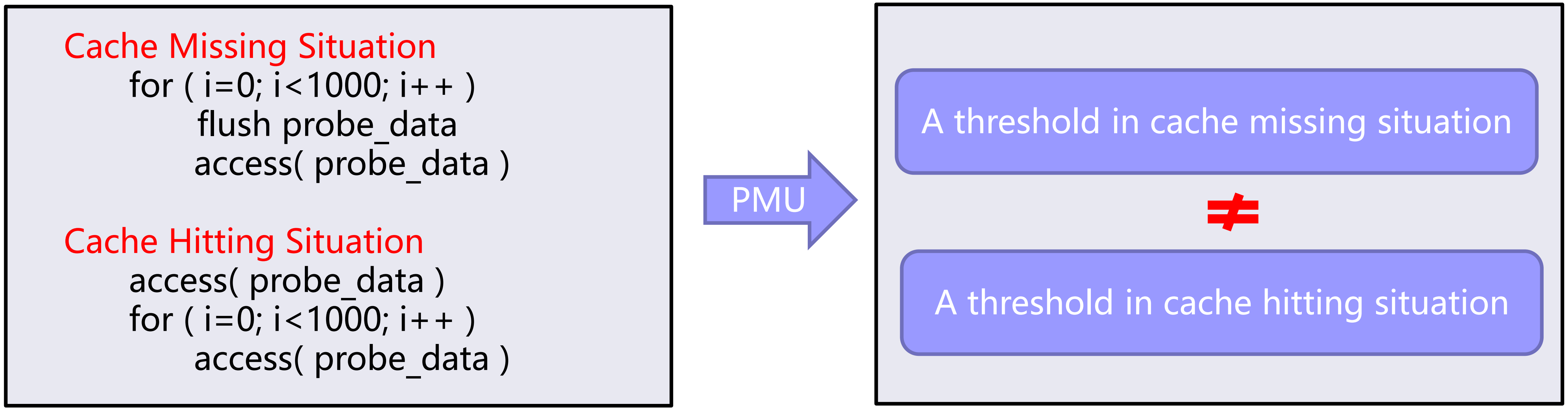
DELL Inspiron 15-7560 laptop

- i5-7200U CPU: vulnerable to Spectre and ZombieLoad attack
- OS: Ubuntu 20.04.3
- Enable Intel SGX
- Kernel version: 4.4.283

Flush+Reload: recover secret data in the transient execution attacks

Prime+Probe: infer the key of an AES performed in SGX

Locate the vulnerable PMU events



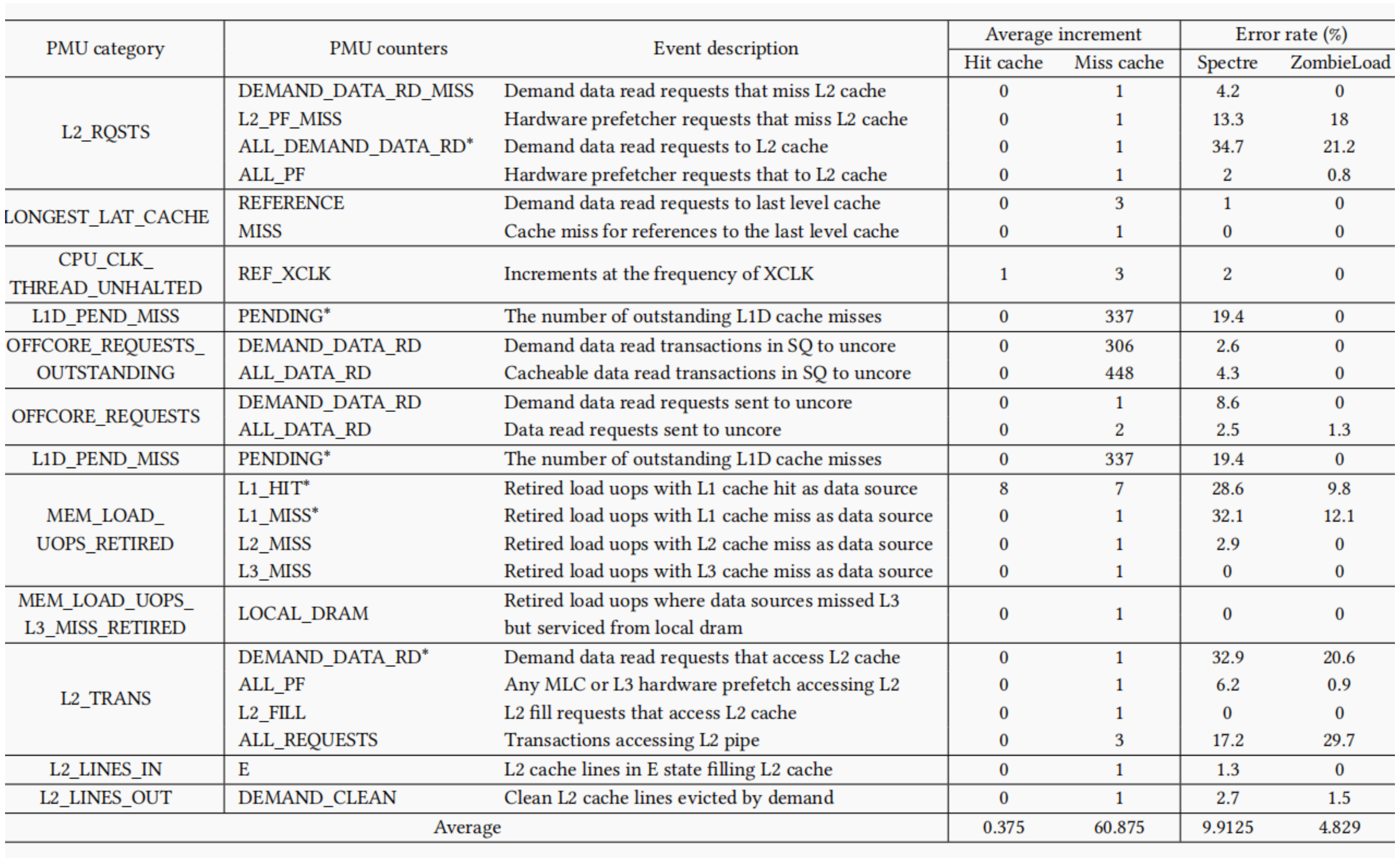
```
Cache Missing Situation
    for ( i=0; i<1000; i++ )
        flush probe_data
        access( probe_data )

Cache Hitting Situation
    access( probe_data )
    for ( i=0; i<1000; i++ )
        access( probe_data )
```

PMU
A threshold in cache missing situation
≠
A threshold in cache hitting situation

Recovery mechanism

- Iterate 1000 times for each PMU counter and each situation
- Calculate a threshold for each PMU counter
- Distinguishes whether the data access hits or misses cache by comparing the increment of a PMU counter with the threshold

Experiment Results



PMU category	PMU counters	Event description	Average increment		Error rate (%)	
			Hit cache	Miss cache	Spectre	ZombieLoad
L2_RQSTS	DEMAND_DATA_RD_MISS	Demand data read requests that miss L2 cache	0	1	4.2	0
	L2_PF_MISS	Hardware prefetcher requests that miss L2 cache	0	1	13.3	18
	ALL_DEMAND_DATA_RD*	Demand data read requests to L2 cache	0	1	34.7	21.2
	ALL_PF	Hardware prefetcher requests that to L2 cache	0	1	2	0.8
LONGEST_LAT_CACHE	REFERENCE	Demand data read requests to last level cache	0	3	1	0
	MISS	Cache miss for references to the last level cache	0	1	0	0
CPU_CLK_ THREAD_UNHALTED	REF_XCLK	Increments at the frequency of XCLK	1	3	2	0
L1D_PEND_MISS	PENDING*	The number of outstanding L1D cache misses	0	337	19.4	0
OFFCORE_REQUESTS_ OUTSTANDING	DEMAND_DATA_RD	Demand data read transactions in SQ to uncore	0	306	2.6	0
	ALL_DATA_RD	Cacheable data read transactions in SQ to uncore	0	448	4.3	0
OFFCORE_REQUESTS	DEMAND_DATA_RD	Demand data read requests sent to uncore	0	1	8.6	0
	ALL_DATA_RD	Data read requests sent to uncore	0	2	2.5	1.3
L1D_PEND_MISS	PENDING*	The number of outstanding L1D cache misses	0	337	19.4	0
MEM_LOAD_ UOPS_RETIRED	L1_HIT*	Retired load uops with L1 cache hit as data source	8	7	28.6	9.8
	L1_MISS*	Retired load uops with L1 cache miss as data source	0	1	32.1	12.1
	L2_MISS	Retired load uops with L2 cache miss as data source	0	1	2.9	0
	L3_MISS	Retired load uops with L3 cache miss as data source	0	1	0	0
MEM_LOAD_UOPS_ L3_MISS_RETIRED	LOCAL_DRAM	Retired load uops where data sources missed L3 but serviced from local dram	0	1	0	0
L2_TRANS	DEMAND_DATA_RD*	Demand data read requests that access L2 cache	0	1	32.9	20.6
	ALL_PF	Any MLC or L3 hardware prefetch accessing L2	0	1	6.2	0.9
	L2_FILL	L2 fill requests that access L2 cache	0	1	0	0
	ALL_REQUESTS	Transactions accessing L2 pipe	0	3	17.2	29.7
L2_LINES_IN	E	L2 cache lines in E state filling L2 cache	0	1	1.3	0
L2_LINES_OUT	DEMAND_CLEAN	Clean L2 cache lines evicted by demand	0	1	2.7	1.5
Average			0.375	60.875	9.9125	4.829

Throughput
- Spectre: 26.02 bps
- ZombieLoad:46.3 bps

Error rate
- Spectre: 9.9125%
- ZombieLoad: 4.829%

24 vulnerable events
- 6 events (marked with "∗ ") recover the AES`s Key
- 24 events recover the secret data in the transient execution attacks

Experiment Results Analysis

Efficiency

- Implement Spectre and ZombieLoad attacks
- Recover the AES`s key

Severity

- Vulnerable PMU events are numerous

Error rate

- Using multiple PMU counters once a time can reduce the error rate

Countermeasures

Hardware

- Disable the vulnerable PMU counters
- Authenticating the users

Software

- Remove PMU from the TCB of TEE
  - Provide a interface to enable/disable PMU through BIOS
  - If clients need to use TEE, they should first disable the PMU
  - Otherwise, PMU can be enabled

Outline

Conclusion

Method

- PMU-Leaker, a new kind of side channel attack

Impleme ntation

- Implement Spectre and ZombieLoad attacks
- Recover the AES`s key

Result

- ➢ 24 events implement transient execution attacks
- 6 events recover the AES`s Key

Thanks

Pengfei Qiu
Beijing University of Posts and Telecommunications

qpf@bupt.edu.cn