EPFL

Revisiting Microarchitectural Side-Channels

by Miro Haller

Responsible:
Prof. Serge Vaudenay
EPFL / LASEC

Supervisor:
Muhammed Fatih Balli
EPFL / LASEC

EPFL

1

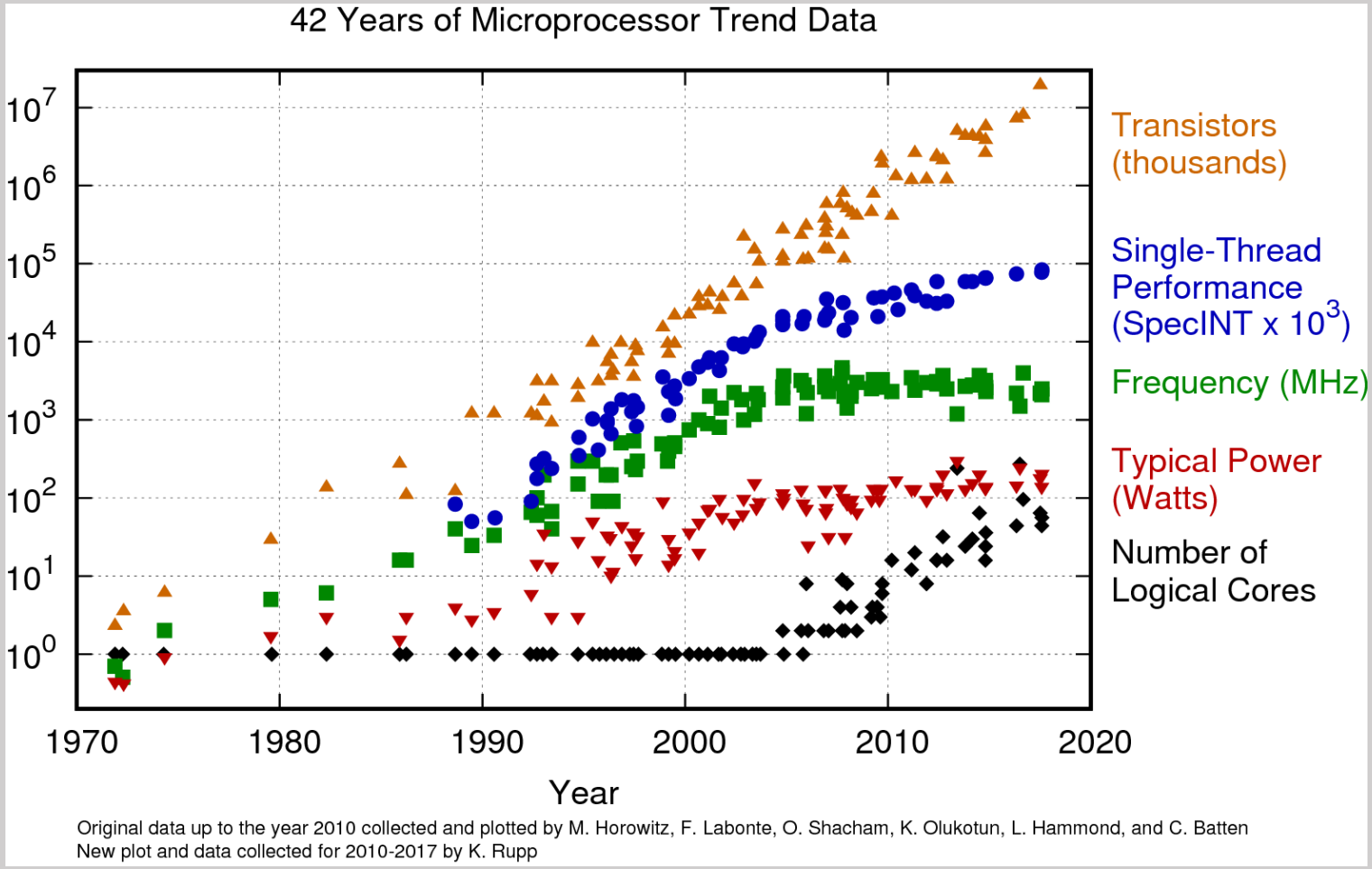
Motivation

Cache Side-Channel History

1
2
3
4
5
6
7
8

- 1996: Kocher publishes first side-channel attacks against PKC systems [1]
- 2002: Page presents theoretical cache attacks [2]
- 2005: Osvik et al. introduce Prime+Probe and implement practical cache attacks [3, 4]
- 2013: Yarom and Falkner apply Flush+Reload to LLC leading to cross-core and cross-VM attacks [5]
- 2016: Gruss et al. introduce the stealthy Flush+Flush cache attack [6]

Increased Complexity



Revisit Side-Channels

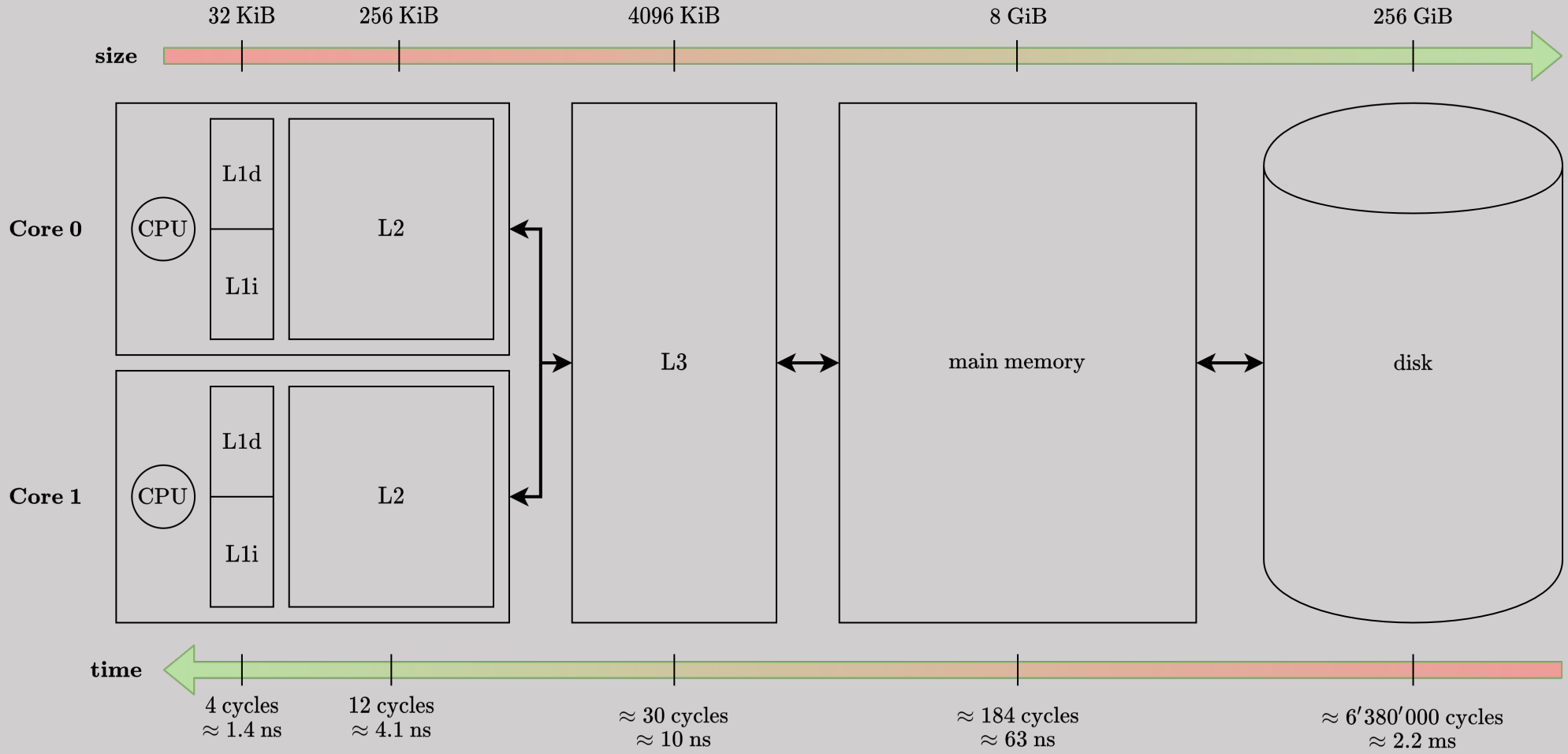
- Processors evolved significantly
- Side-Channels likely to persist
- Empirical approach
- Applicable to recently introduced algorithms?
  - Argon2d

EPFL

2

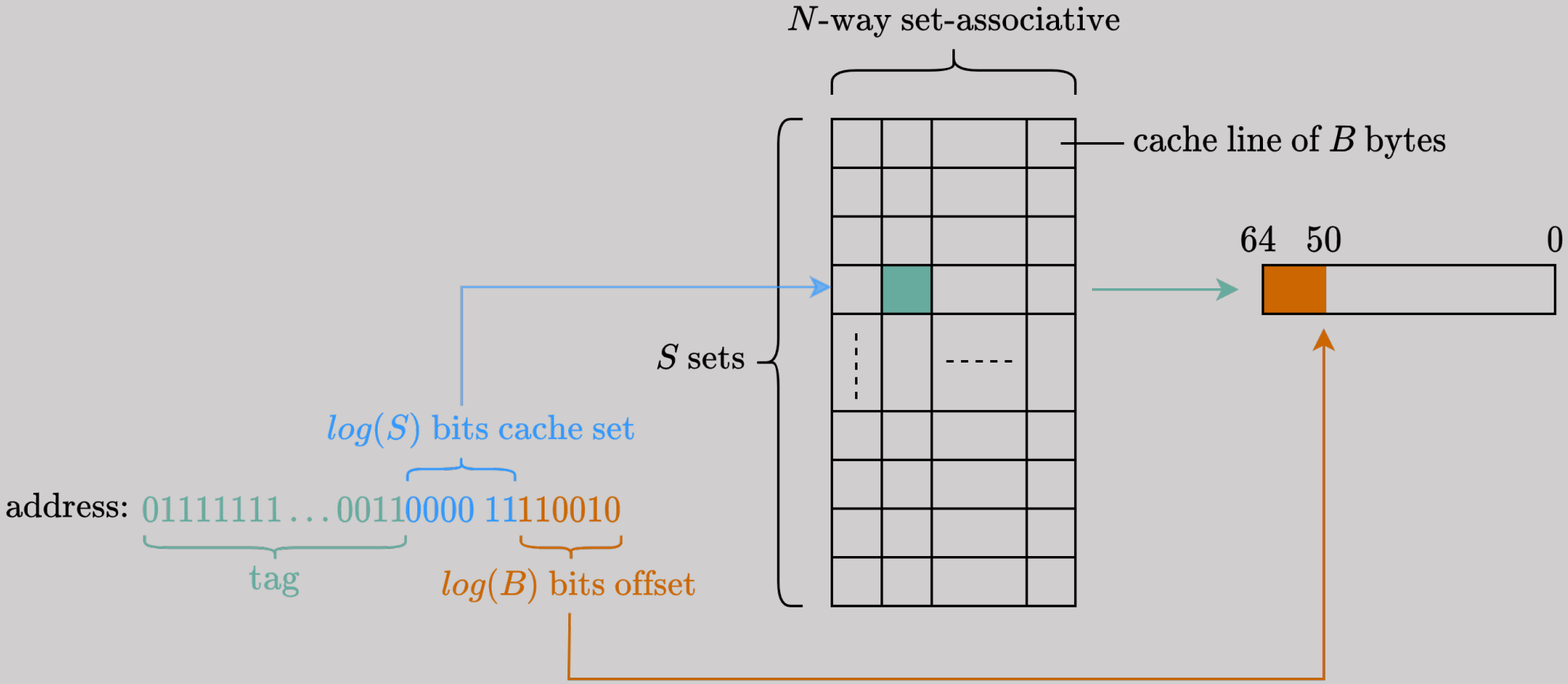
Cache Architecture

Memory Subsystem



concrete numbers for Ivy Bridge with Intel® Core™ i7-3520M CPU @ 2.90GHz

Cache Terminology and Addressing

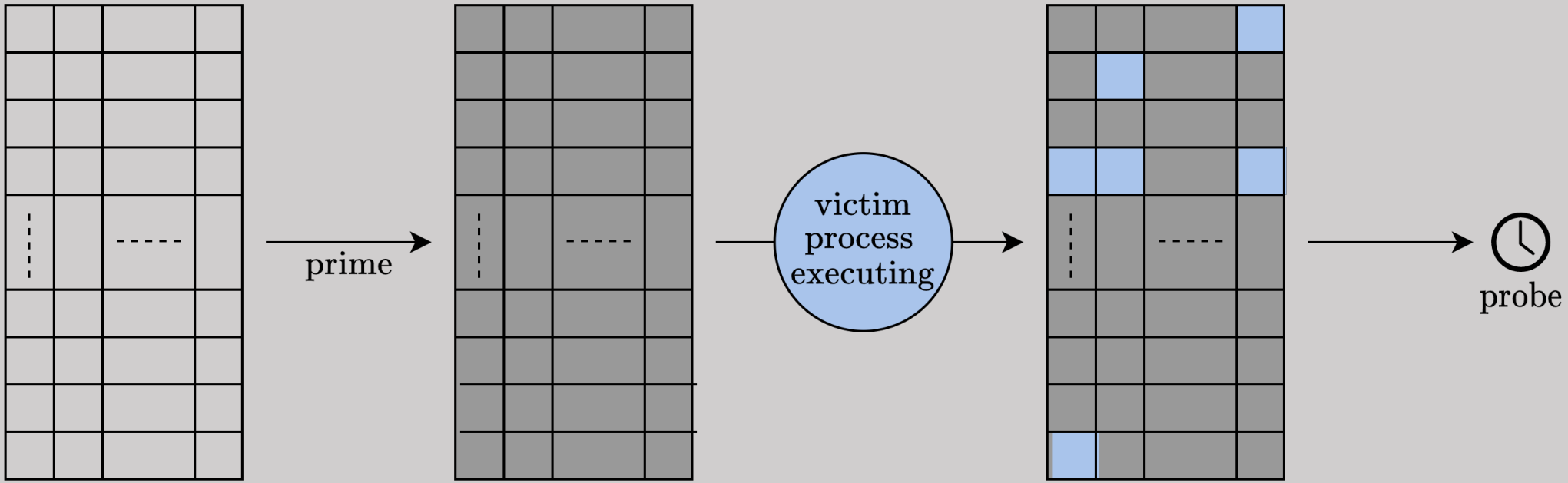


EPFL

3

Prime+Probe

Prime+Probe in Theory



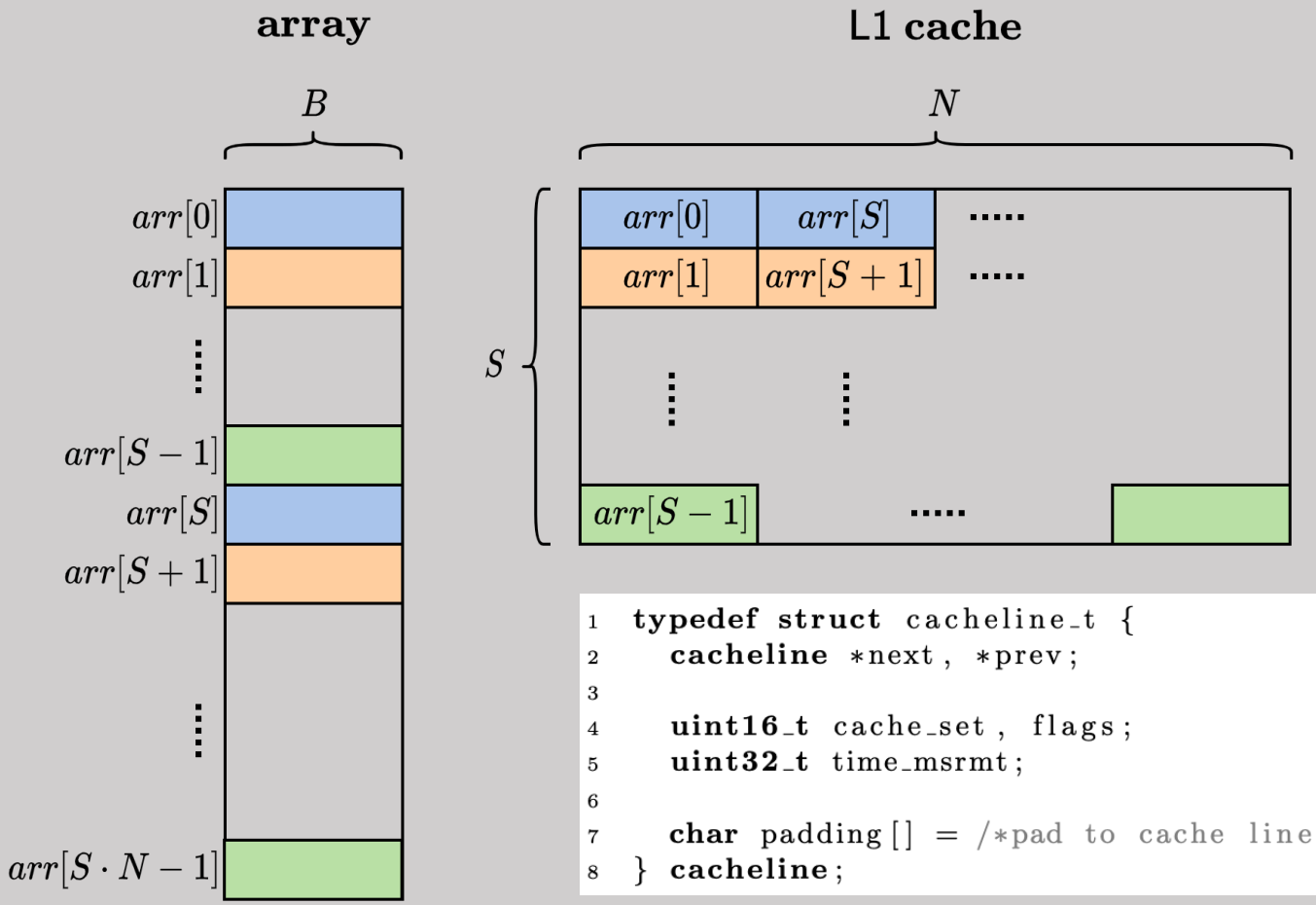
Prime+Probe in Practice I

- Precise time measurements
  - Out-of-order execution
    - Fencing
  - Dynamic frequency scaling
    - CPU governor
    - BIOS
    - Busy loop heuristic

```
1  cpuid
2  rdtsc
3  /* store timestamp */
4  /* measured code */
5  rdtscp
6  /* store timestamp */
7  cpuid
```

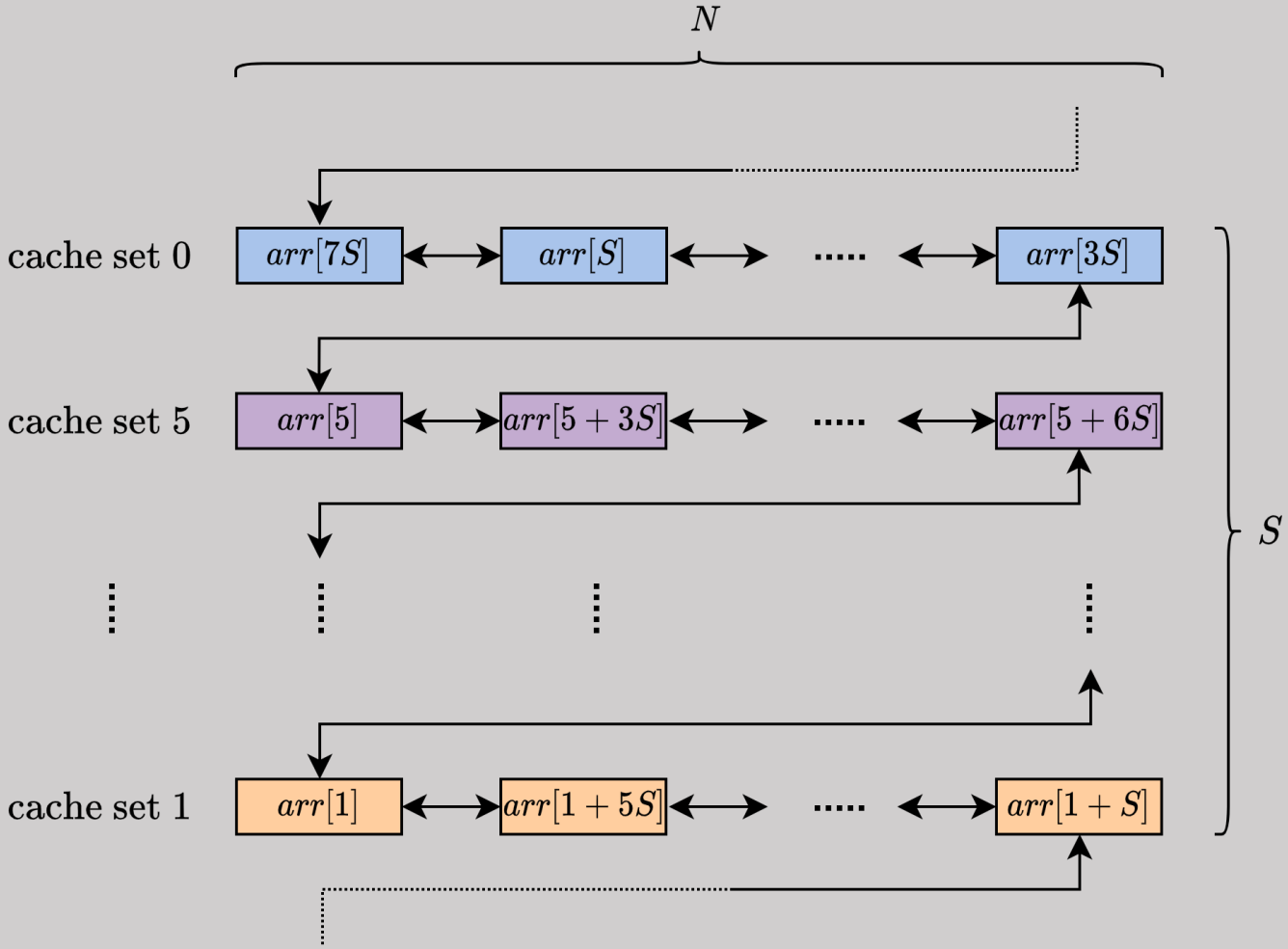
Code benchmarking following the recommendation of Intel's benchmarking white paper [7]

Prime+Probe in Practice II

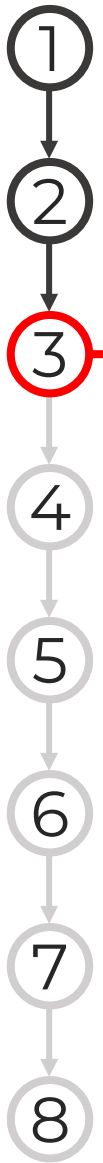



```
1  typedef struct cacheline_t {
2    cacheline *next, *prev;
3
4    uint16_t cache_set, flags;
5    uint32_t time_msrmt;
6
7    char padding[] = /*pad to cache line size*/;
8  } cacheline;
```

Prime+Probe in Practice III

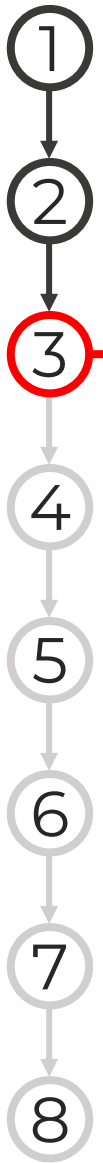


Prime+Probe in Practice IV



- Further issues
  - Compiler
  - Function call overhead
  - Approximate LRU
  - Write buffers and LFBs
  - Branch prediction

Lessons Learned



- Simplicity is key
  - Reduce noise
  - Synchronous measurement
  - Increase complexity step-by-step
- The data structure is essential
  - Minimize cache pollution
  - Avoid hardware prefetching
  - Reduce overhead

EPFL

4

Applications I – OpenSSL AES-CBC

AES-CBC with T-Tables

- T-tables Te_0, Te_1, Te_2, Te_3 combine the SubBytes, ShiftRows, and MixColumns transformations into lookup tables
- Set of bytes $\mathbb{B} = \{0,1\}^8$, round $0 \leq r \leq 10$
- Plaintext block $P = p_0 || \, p_1 || \, \ldots || p_{15}$, key $\mathrm{K} = k_0 || \, k_1 || \, \ldots || k_{15}$
- Key schedule produces key matrices $K^{(r)} \in \mathbb{B}^{4\times4}$ for each round
- Let $A^{(r)} \in \mathbb{B}^{4\times4}$ be the input to the T-tables in round r
- $A_{i,j}^{(0)} = p_l \oplus k_l$ for $l = i + 4j, 0 \leq i, j < 4$
- $A_{:,j}^{(r+1)} = Te_0\left[A_{0,j}^{(r)}\right] \oplus Te_1\left[A_{1,j+1}^{(r)}\right] \oplus Te_2\left[A_{2,j+2}^{(r)}\right] \oplus Te_3\left[A_{3,j+3}^{(r)}\right] \oplus K_{:,j}^{(r)}$ for $0 \leq r < 9$
- Last round $r = 9$ uses another table Te_4

CPA attack on $r = 0$ Table Lookups

- First round table lookups depend on key and plaintext:

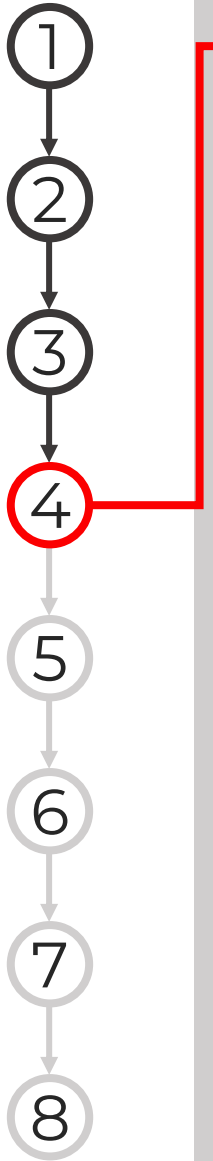
$$Te_t[p_l \oplus k_l] \text{ for } l = t + 4s, 0 \leq t, s < 4$$

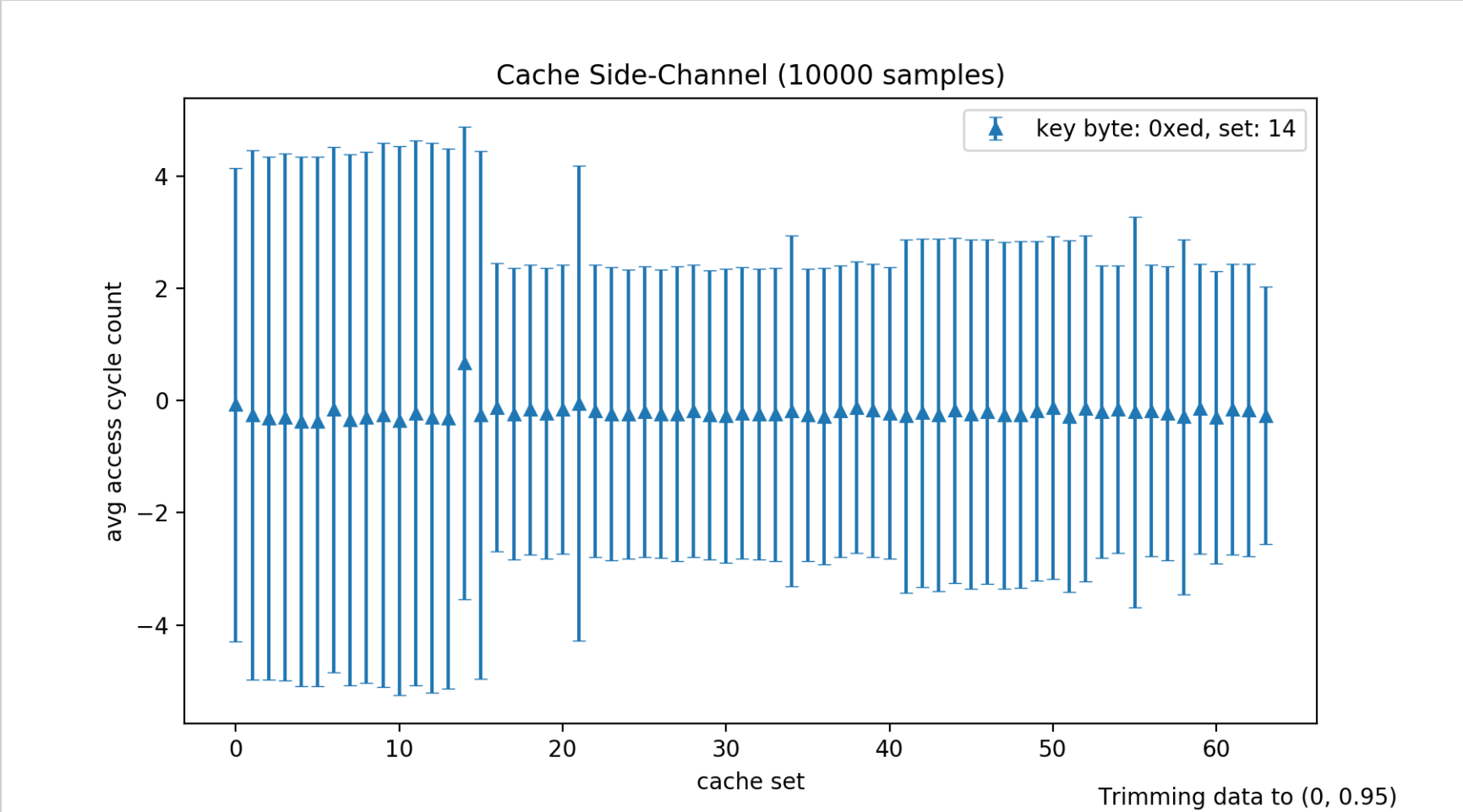
- Lookup table entries are 4 bytes
- Cache set accesses leak 4 bits of k_l:

$$Te_t[a] \Longleftrightarrow *(Te_t + a) \Longleftrightarrow \begin{matrix} ptr_{Te_t} \\ + \ a \cdot 4\text{B} \end{matrix} \Longleftrightarrow \begin{matrix} \\ + \end{matrix} \begin{matrix} b_{63}b_{62}\ldots b_{12} & \overbrace{b_{11}b_{10}b_9b_8b_7b_6}^{\text{cache set}} & \overbrace{b_5b_4b_3b_2b_1b_0}^{\text{offset}} \\ & a_7a_6a_5a_4 & a_3a_2a_1a_000 \end{matrix}$$

- Extendable to recover half of the entire key

Cache Observation of OpenSSL CPA





Conclusion

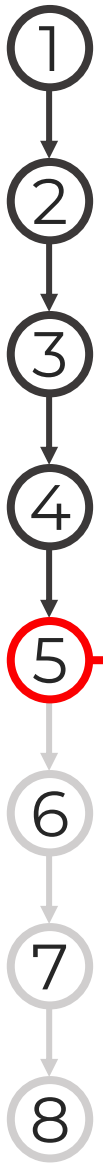
- CPA on AES-CBC of OpenSSL-0.9.8 is reproducible on contemporary hardware
- OpenSSL-1.1.1 still includes this code
- But unlikely to be executed today
  - AES-NI

EPFL

5

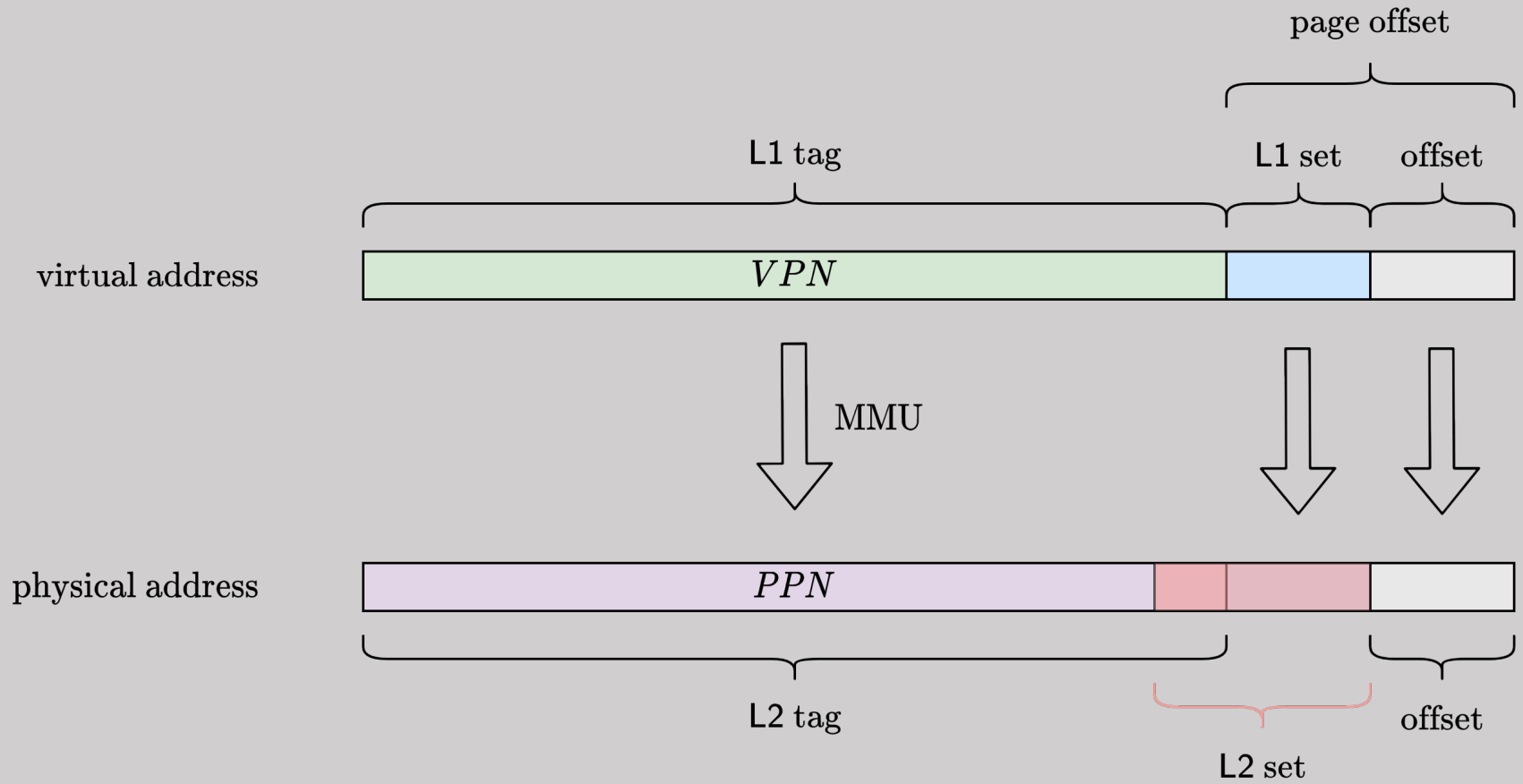
Prime+Probe on Physically Indexed Caches

Why Target Physically Indexed Caches?

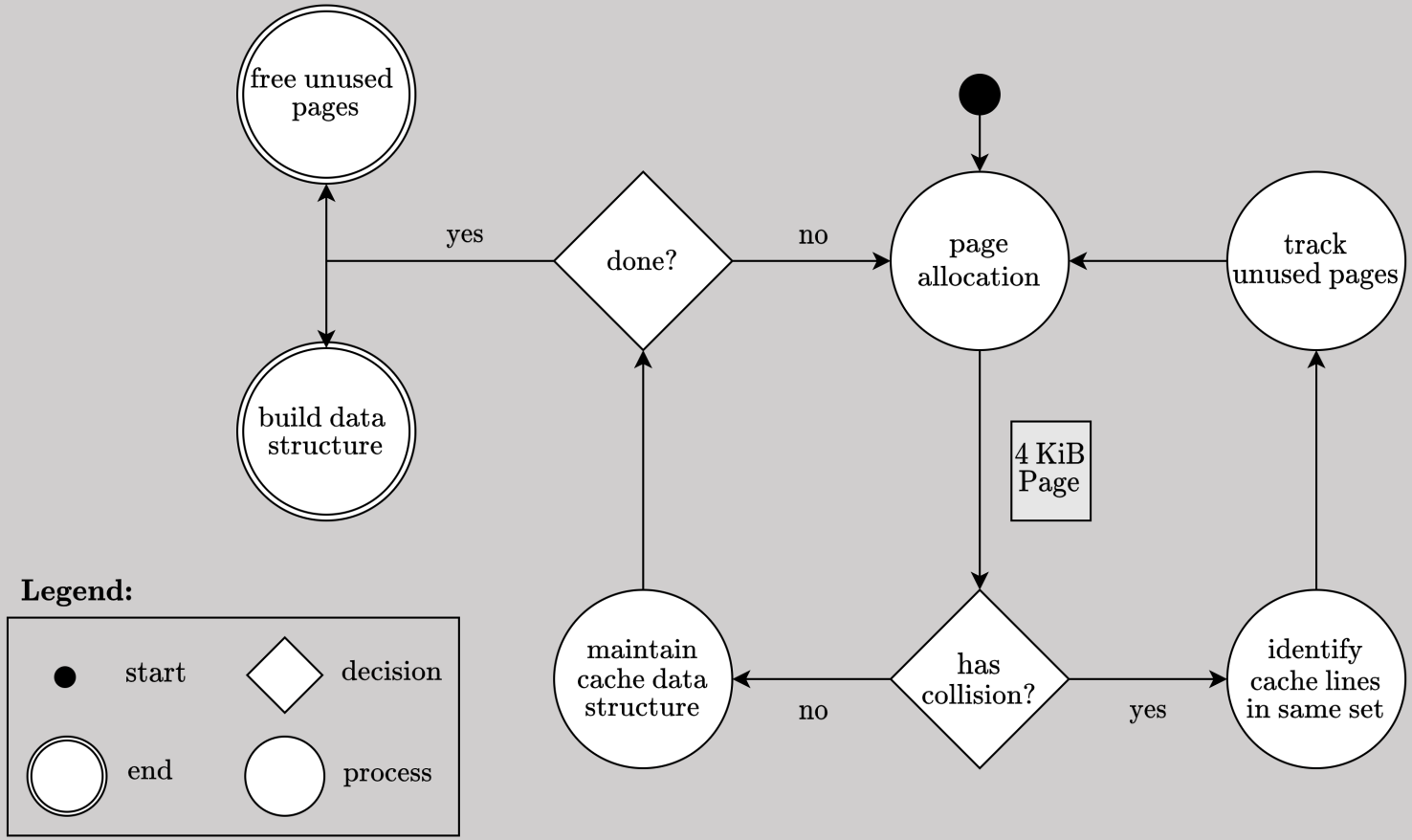


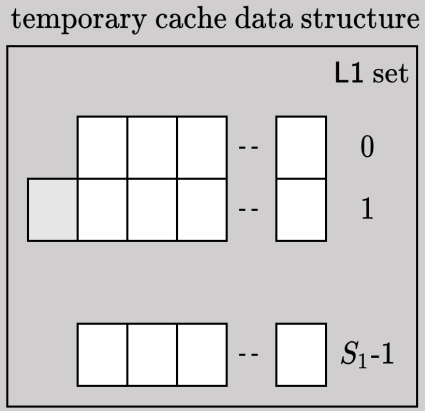
- Common architecture:
  - L1 virtually indexed
  - L2, L3 physically indexed
- Prime+Probe on L1 or Flush+Reload on L3
- Advantages of Prime+Probe on L2:
  - Larger than L1
  - No shared addresses needed
  - Higher cache miss penalty than for L1
- Challenge:
  - Building cache data structure

Virtual to Physical Address Translation

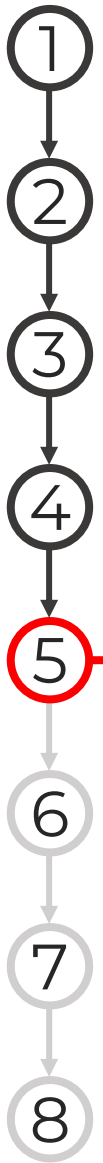


Building the Data Structure





Limitations



- Limitations:
  - Privileged version requires CAP_SYS_ADMIN capability
  - Unprivileged version is slow
  - Precise L1 sets, permutation of L2 sets

6

Applications II – Argon2d

Argon2 [11]

- Memory hard password hash function
- Two variants:
  - Argon2d:
    - Data-dependent memory accesses
    - Faster
    - Suitable for cryptocurrencies
  - Argon2i:
    - Data-independent memory accesses
    - Preferred for password hashing

Argon2 [11]

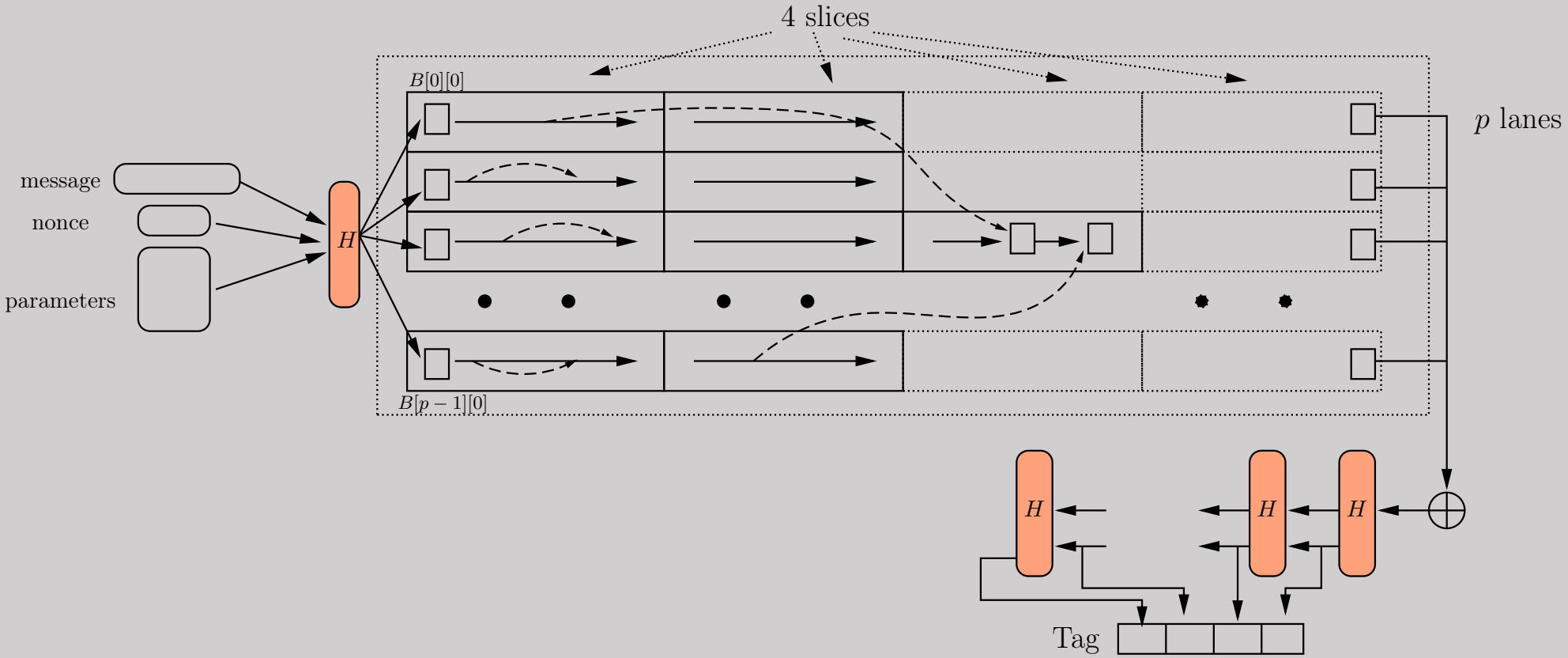
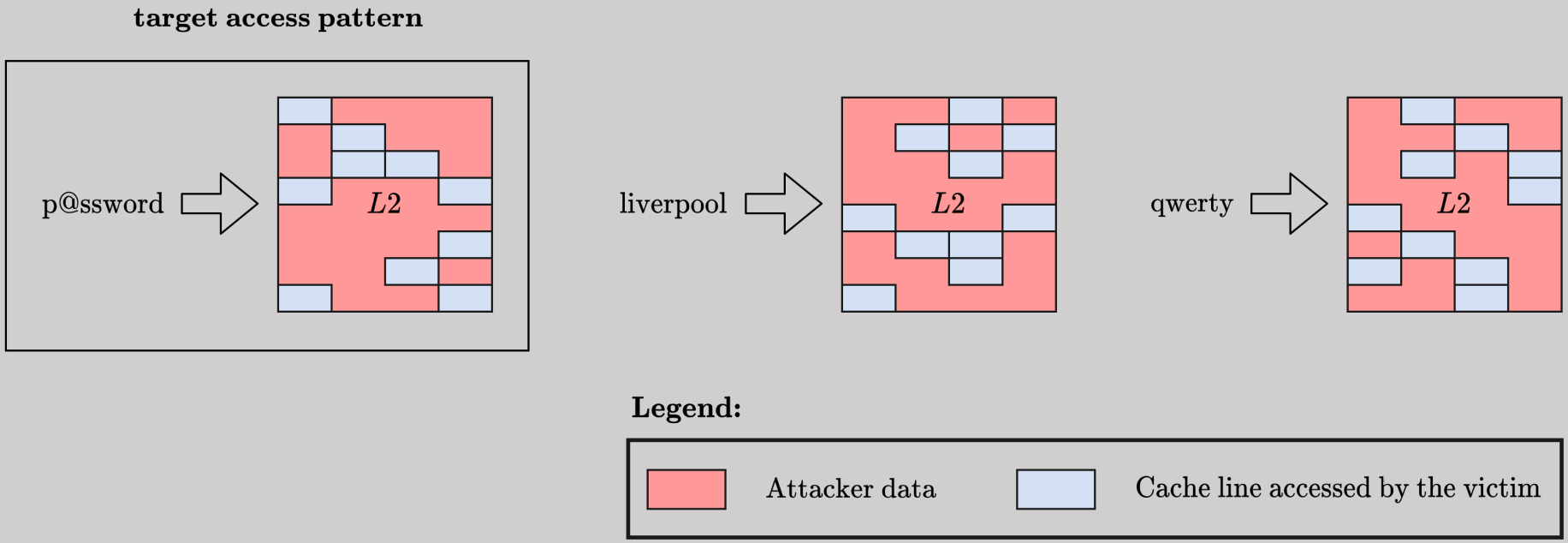


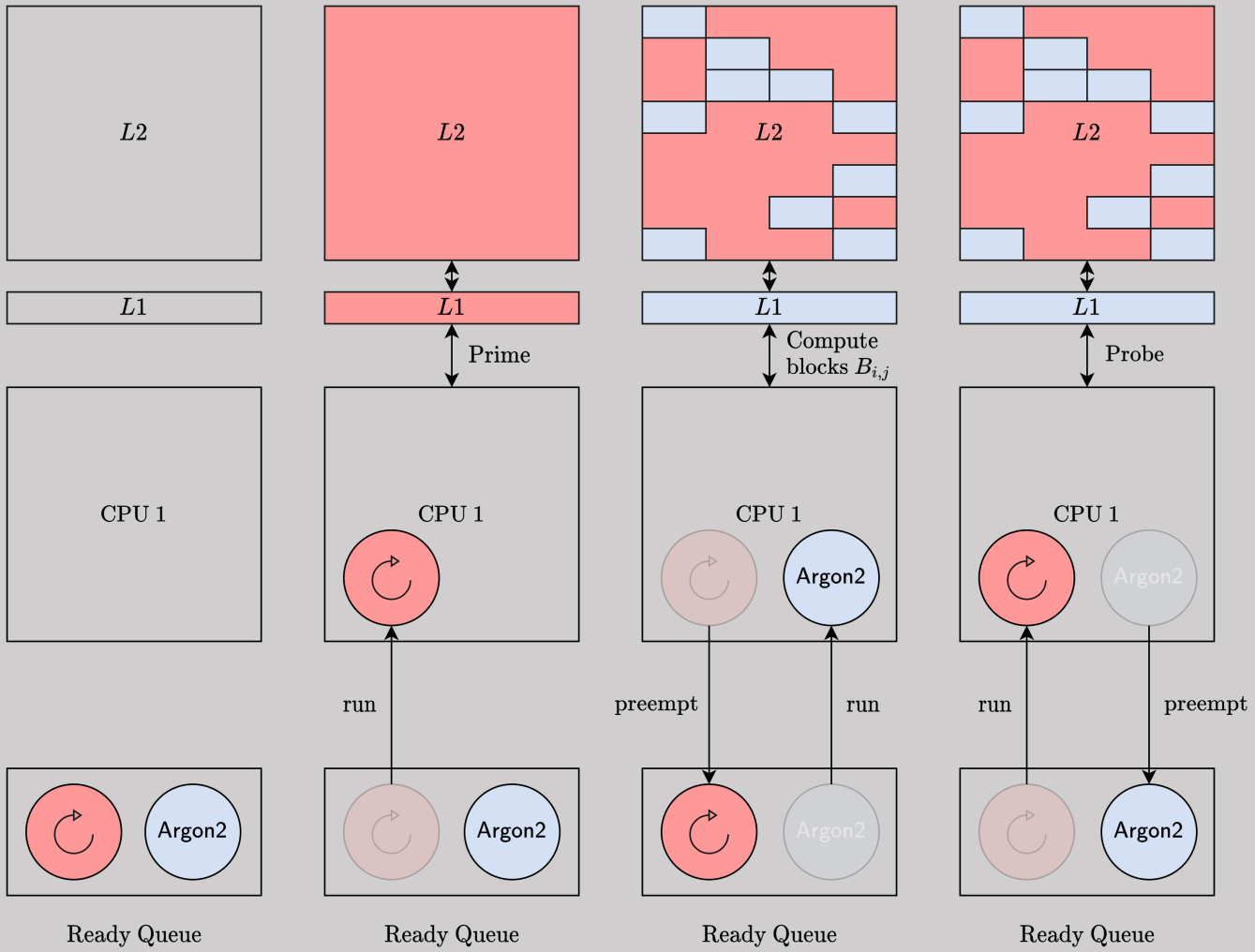
Figure 2 from the Argon2 paper by Alex Biryukov, Daniel Dinu, and Dmitry Khovratovich [11].

Motivation

- Password-dependent cache access pattern
- Efficient password cracking through early abortion
- Depends on cache observation accuracy



Argon2d Observation Granularity I



Legend:

- Attacker process looping Prime+Probe
- Argon2 — Victim process computing Argon2 hash
- Cache line accessed by the attacker
- Cache line accessed by the victim

Argon2d Observation Granularity II

- Empirical results:

Ratio of Target Sets	$\mu^{\mathcal{A}}_{C_{P+P}}$	$\sigma^{\mathcal{A}}_{C_{P+P}}$	$\mu^{\mathcal{V}}_{C_{blocks}}$	$\sigma^{\mathcal{V}}_{C_{blocks}}$
512/512	34.4	2.3	6517.7	1460.9
32/512	595.6	41.9	6520.9	1445.6

- Results:
  - Independent of the attacker
  - Victim accesses approx. 25× the number of cache lines in L2
- Further research:
  - Required granularity to distinguish access patterns
  - Slow down victim

EPFL

7

CacheSC

Conclusion

- Prime+Probe on L1
  - Classic OpenSSL AES-CBC CPA on contemporary hardware
- Prime+Probe on L2
  - Cache observation granularity of Argon2d
- Insights into cache attack data structure design
- CacheSC

GitHub

"Library for Prime+Probe cache side-channel attacks on L1 and L2"

8

References

References

1. Paul C. Kocher. Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems. In Neal Koblitz, editor, Advances in Cryptology — CRYPTO '96, pages 104–113, Berlin, Heidelberg, 1996. Springer Berlin Heidelberg.
2. Dan Page. Theoretical use of cache memory as a cryptanalytic side-channel. IACR Cryptology ePrint Archive, 2002:169, 01 2002.
3. Dag Arne Osvik, Adi Shamir, and Eran Tromer. Cache Attacks and Countermeasures: The Case of AES. January 2005.
4. Daniel J. Bernstein. Cache-timing attacks on AES. 2005.
5. Yuval Yarom and Katrina Falkner. FLUSH+RELOAD: A High Resolution, Low Noise, L3 Cache Side-Channel Attack. In 23rd USENIX Security Symposium (USENIX Security 14), pages 719–732, San Diego, CA, August 2014. USENIX Association.
6. Daniel Gruss, Clémentine Maurice, Klaus Wagner, and Stefan Mangard. Flush+ Flush: a fast and stealthy cache attack. In International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment, pages 279–299. Springer, 2016.
7. Gabriele Paoloni. How to Benchmark Code Execution Times on Intel® IA-32 and IA-64 Instruction Set Architectures. https://www.intel.com/content/dam/www/public/us/en/documents/white-papers/ia-32-ia-64-benchmark-code-execution-paper.pdf, 2010-09.

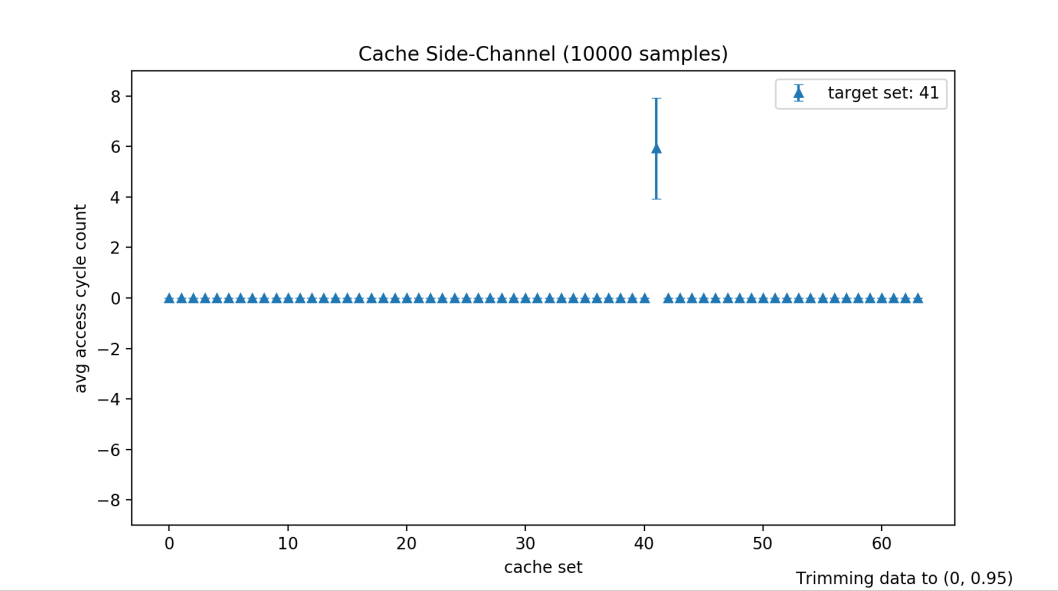
References

8. Alex Biryukov, Daniel Dinu, and Dmitry Khovratovich. Argon2: new generation of memory-hard functions for password hashing and other applications. In 2016 IEEE European Symposium on Security and Privacy, pages 292–302. IEEE, 2016.

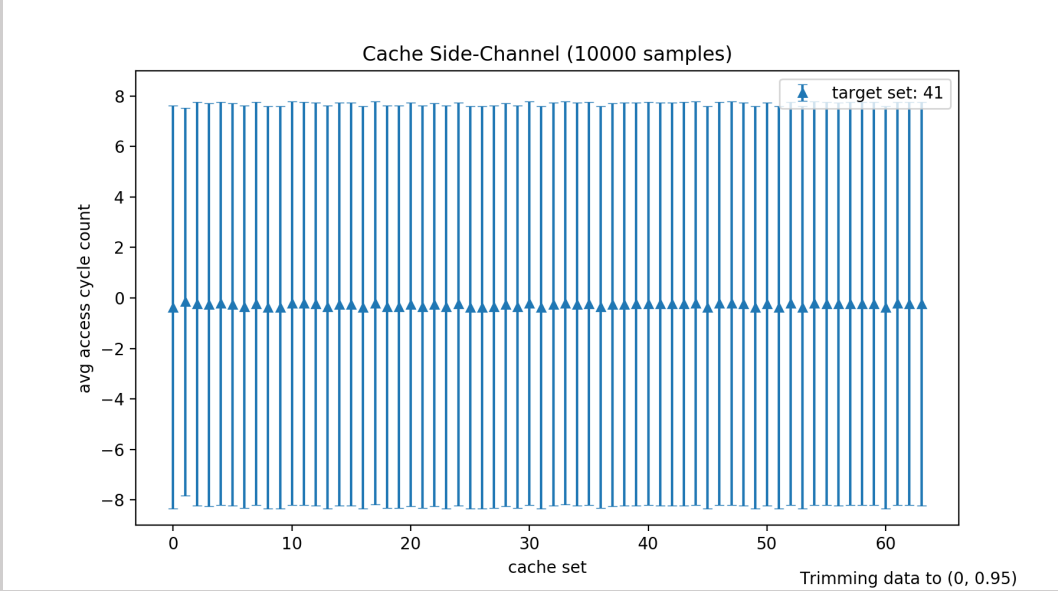
9

Backup Slides

Single Eviction

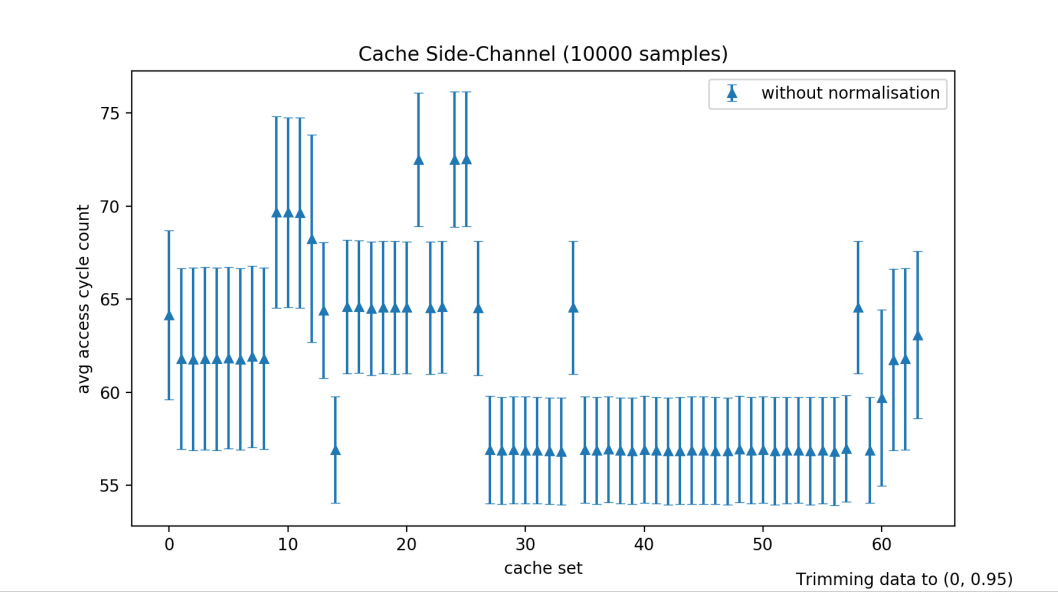


Measure all cache lines in the same set simultaneously

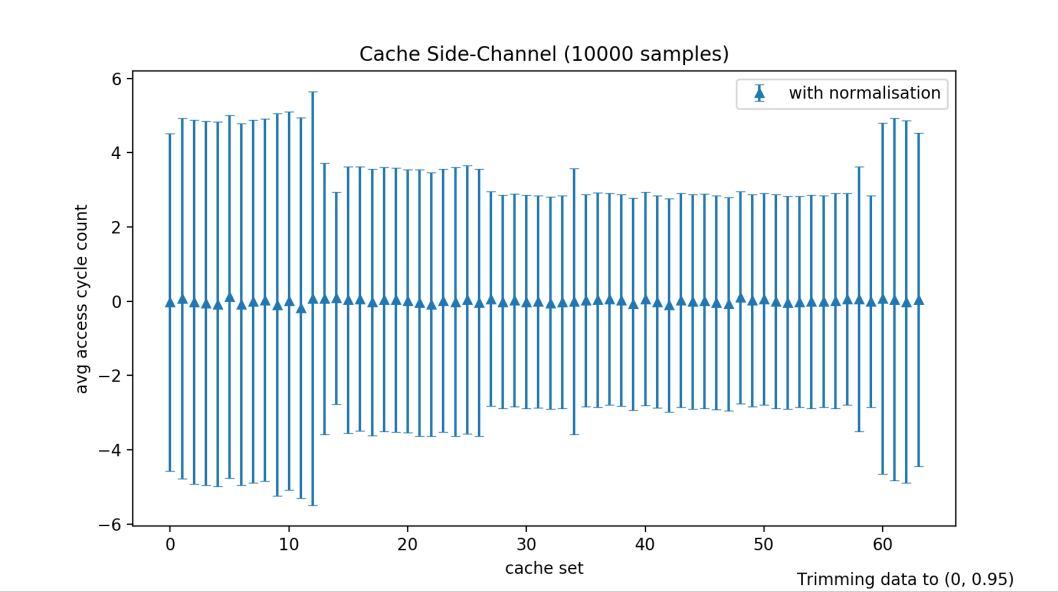


Measure all cache lines individually and sum those of the same cache set

Normalization

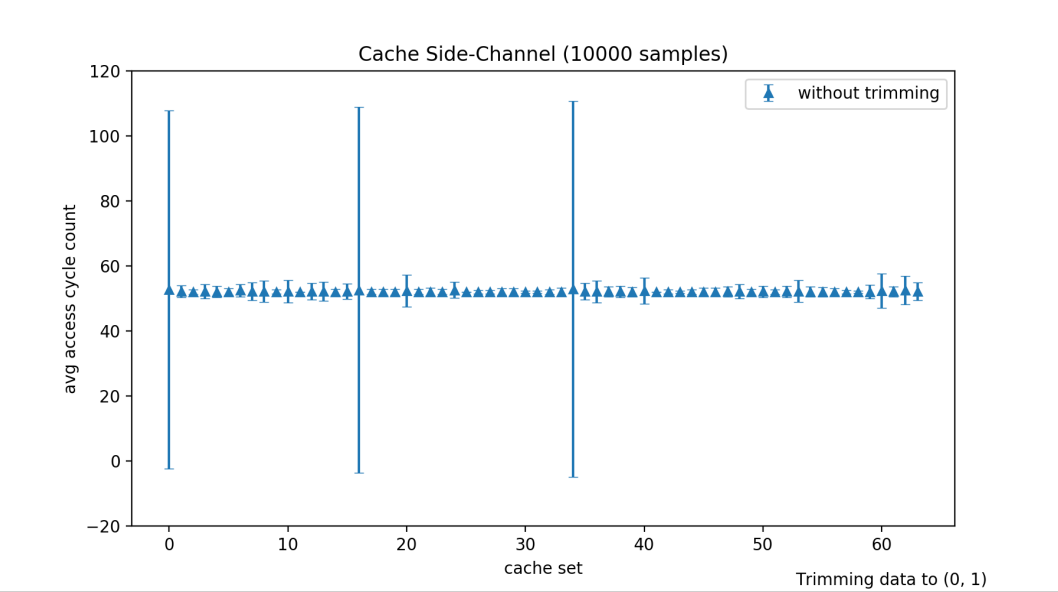


OpenSSL AES-CBC library call
without normalization

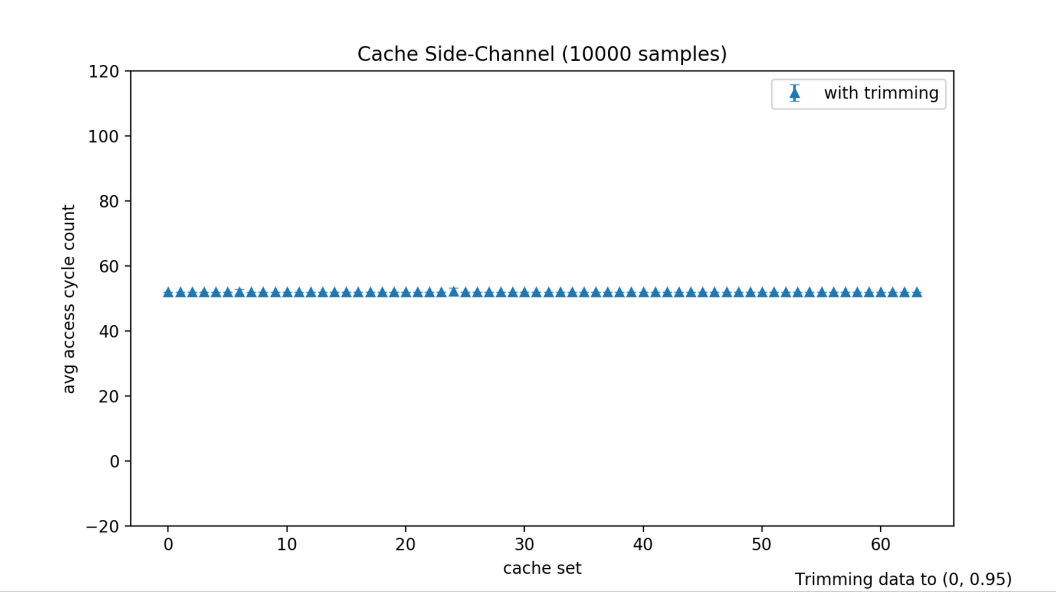


OpenSSL AES-CBC library call
with normalization

Trimming



Prime+Probe (without any evictions) without trimming



Prime+Probe (without any evictions) with trimming

OpenSSL CPA Accuracy