VIRGINIA TECH

CSET '19

Comparative Measurement of Cache Configurations' Impacts on Cache Timing Side-Channel Attacks

Xiaodong Yu, *Ya Xiao, Kirk W. Cameron, Danfeng (Daphne) Yao*

Dept. of Computer Science, Virginia Tech

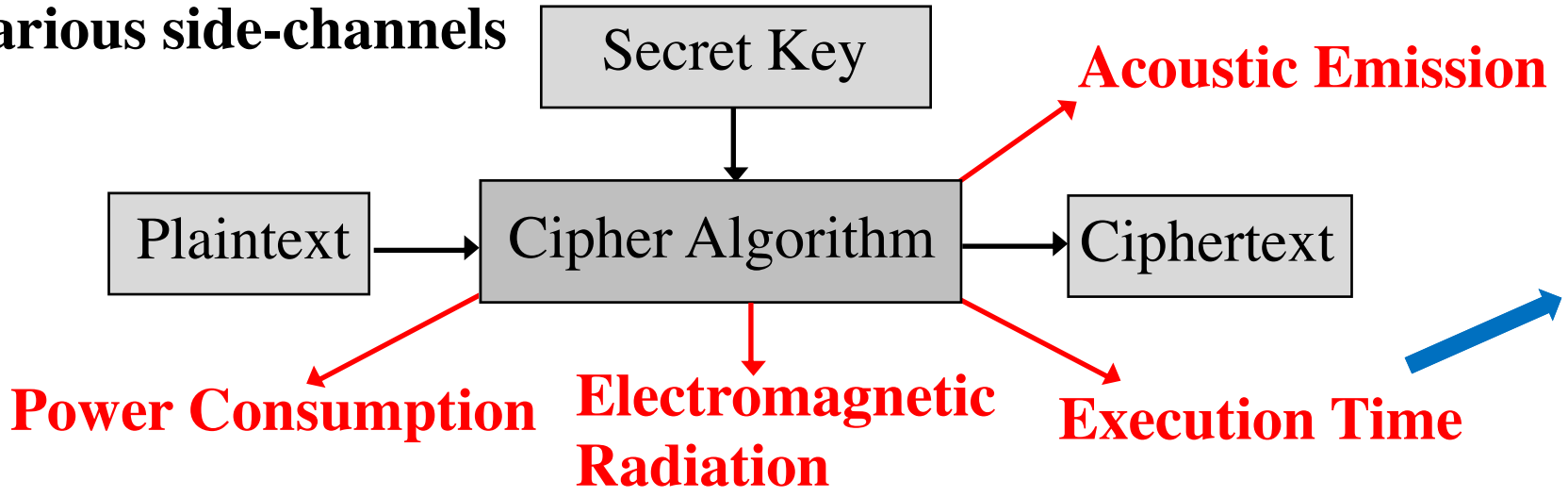
Special Thanks to

- the whole *VarSys* project team:

Thomas Lux, Bo Li, Jon Bernard, Chandler Jearls, Li Xu, Tyler Chang, Prof. Yili Hong, Prof. Layne Watson, Prof. Godmar Back, and Prof. Margaret Ellis

Cache Side-Channel Attacks are Real Dangers

Various side-channels



Cache Attacks:
Guess the secret keys by exploiting the time differences between the cache-hits and cache-misses

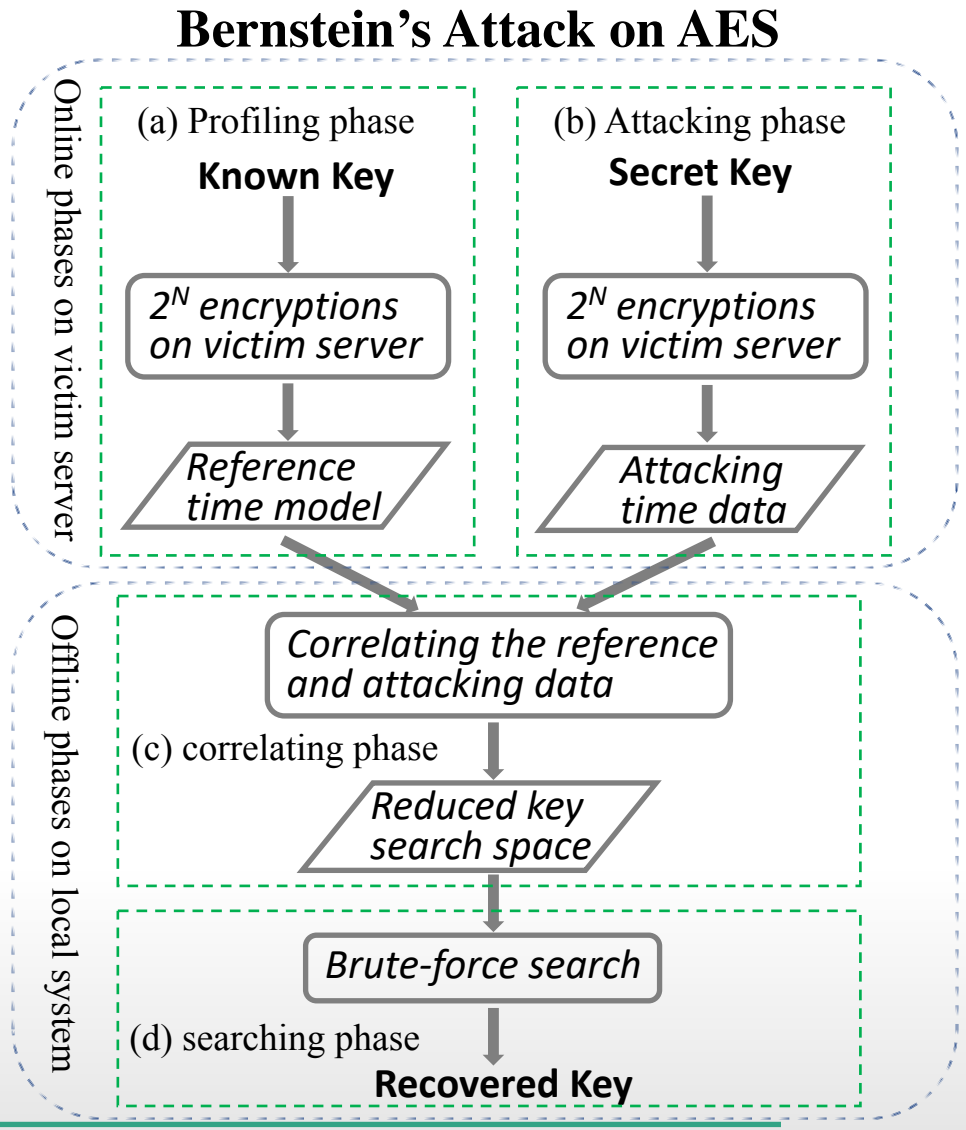
Cache Side-Channel Attacks are Practical and Severe

Both attacks break the application isolations by leveraging the cache side-channel as well as other instruction processing vulnerabilities

Targeted Problem

How do cache configurations influence the performance of cache side-channel attacks?

Cache Configurations' Impact on the Time-Driven Side-Channel Attacks



Time-driven attacks solely rely on the time differences between cache-hits and cache-misses; more cache-misses → easier attacks

Cache configurations can impact the time-driven attacks; but it's unclear **HOW**

Two difficulties in comparative measurements:

- There is no quantifiable metric for the attacks
- There is no configurable caches in commodity CPUs

Our Design: A Quantifiable Metric for Time-driven Cache Attacks

The conventional success-fail binary metric cannot support the comparative measurements

Equivalent Key Length (EKL): a normalized metric to represent the key search space

$$EKL = 1 - \frac{\sum_{k=0}^{n} \log_2 v_k}{8n}$$

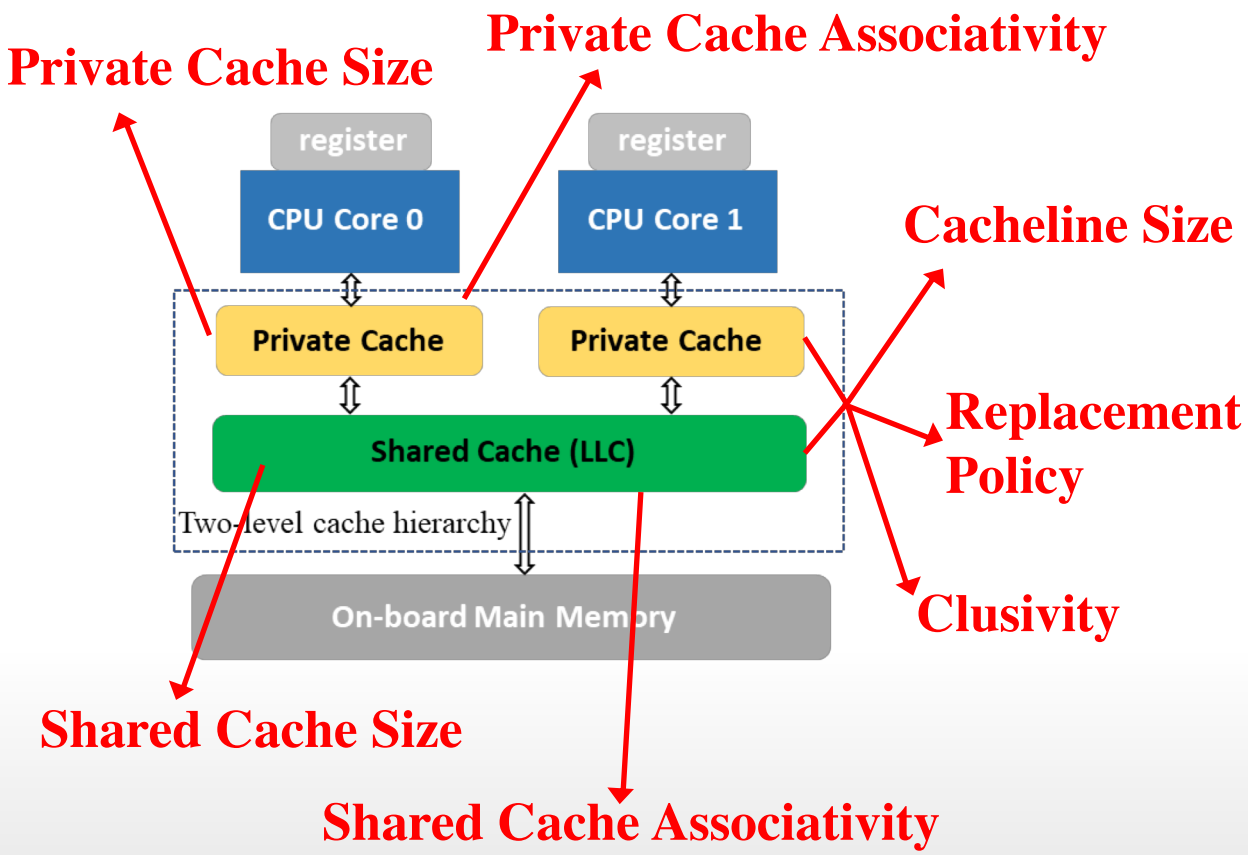
$EKL \in [0,1]$, n is the length of the key (16-bytes in our measurements), v_k is the number of candidates for the k-th key byte, where $k \in [0,n-1]$

EKL=0 → original search space; *EKL=1* → fully revealed secret key
It is unnecessary to achieve EKL=1; Practically, EKL=0.8 can achieve a good balance of the measurement cost and the brute-force search cost

The *success rate* of the attacks: *EKL*/number of encryptions

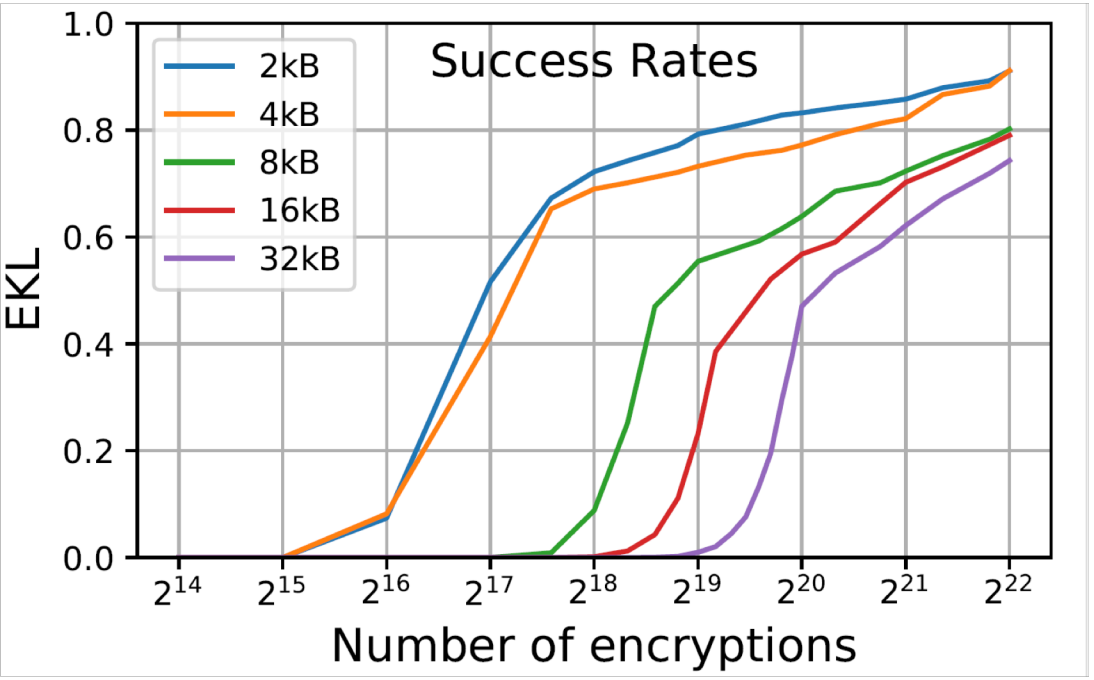
Our Design: use GEM5 to Emulate the Configurable Caches

GEM5 is a modular platform that has been widely used in computer architecture research community; we use GEM5 to cycle-accurately emulate the systems with configurable cache



1. *Private Cache Size (**PCS**)*: 2KB, 4KB, 8KB, 16KB, 32KB
2. *Private Cache Associativity (**PCA**)*: 2-way, 4-way, 8-way, 16-way, 32-way
3. *Shared Cache Size (**SCS**)*: 2MB, 4MB, 8MB, 16MB, 32MB
4. *Shared Cache Associativity (**SCA**)* : 2-way, 4-way, 8-way, 16-way, 32-way
5. *Cacheline Size (**CLS**)*: 32Bytes, 64B, 128B
6. *Replacement Policy (**RP**)*: RANDOM, FIFO, LRU, LFU
7. *Cache Clusivity (**CC**)*: inclusive, exclusive

Measurement Results: Private Cache Size (PCS)



X-axis: the number of encryptions that the attacker conducted
Y-axis: the equivalent key length (EKL)

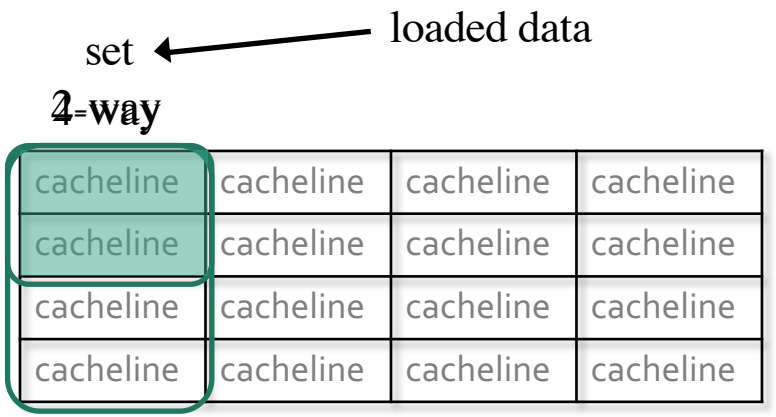
Theoretical: larger PCS leads to more difficult attacks; after 4kB PCS, the attacks are impossible

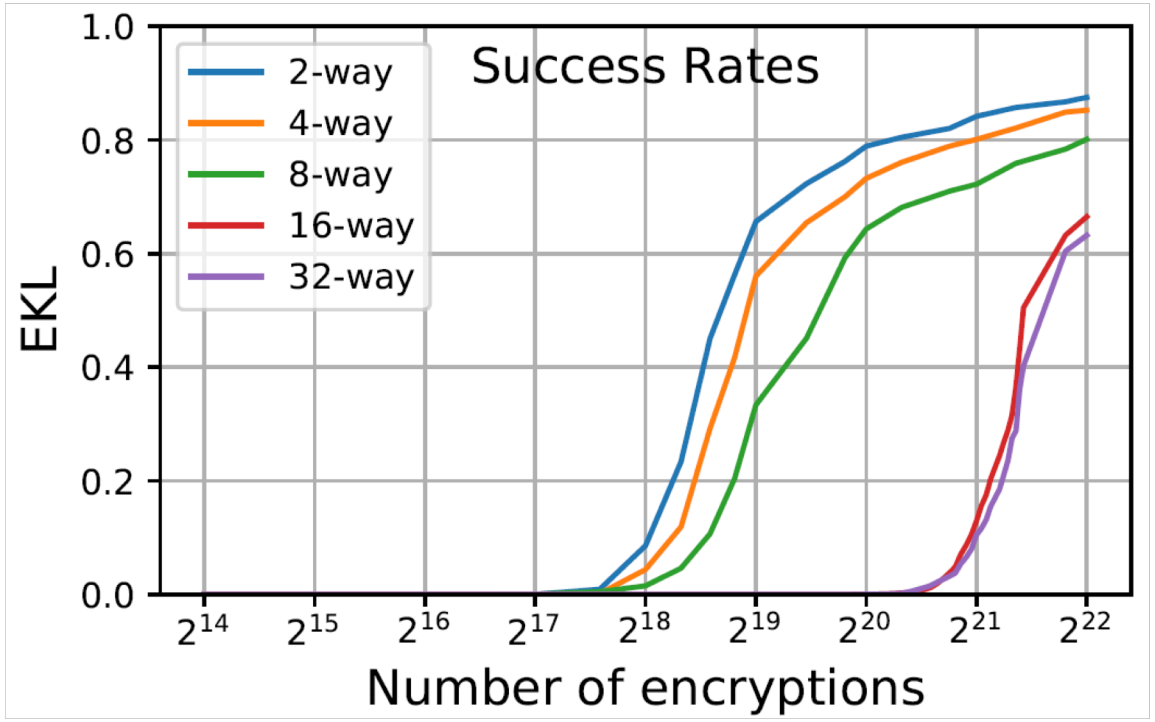
Fact: after 4kB PCS, although much harder, the attacks still can succeed

Reason: AES computation itself and the system operations can kick some lookup table entries out of the private cache

Measurement Results: Private Cache Associativity (PCA)

Basic cache unit is not Byte, but the cacheline





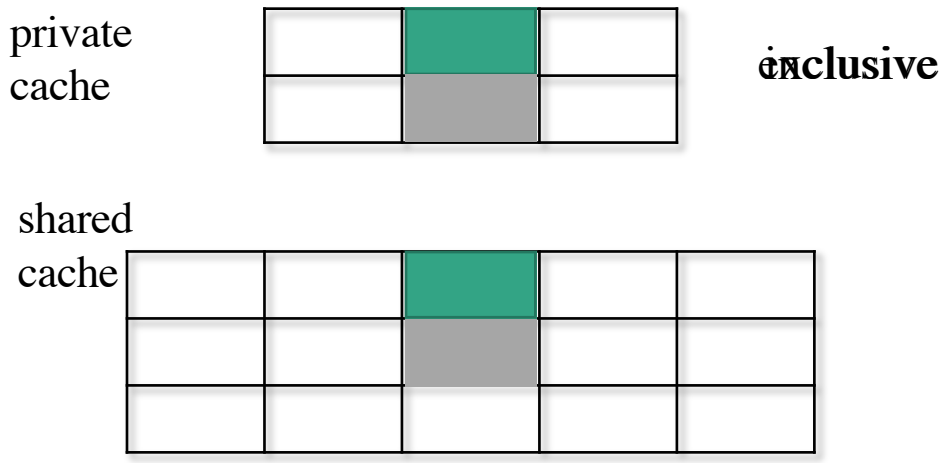
Theoretical: larger PCA leads to more difficult attacks

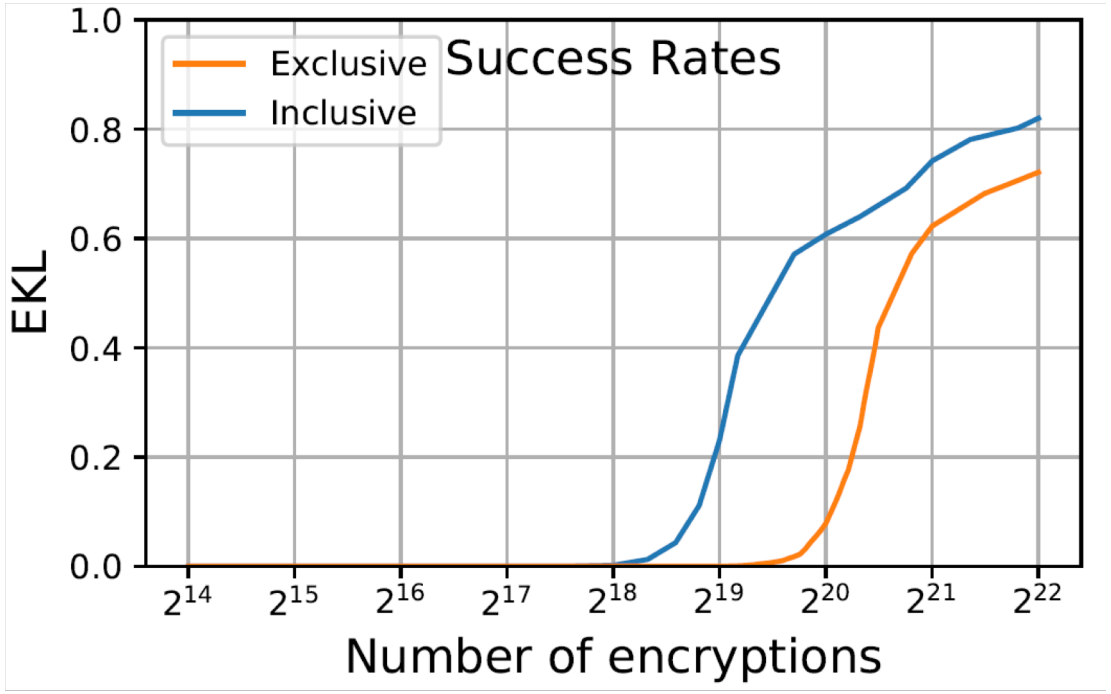
Fact: after 8-way PCA, the attacks get significant harder to succeed

Reason: after 8-way PCA, loaded entry mostly can find an appropriate place in the set without flushing the next-read data

Measurement Results: Cache Clusivity (CC)

Clusivity describes the consistency policy between private and shared cache





Theoretical: inclusive policy results in less overall cache-miss penalties, hence harder attacks

Fact: exclusive policy leads to harder attacks

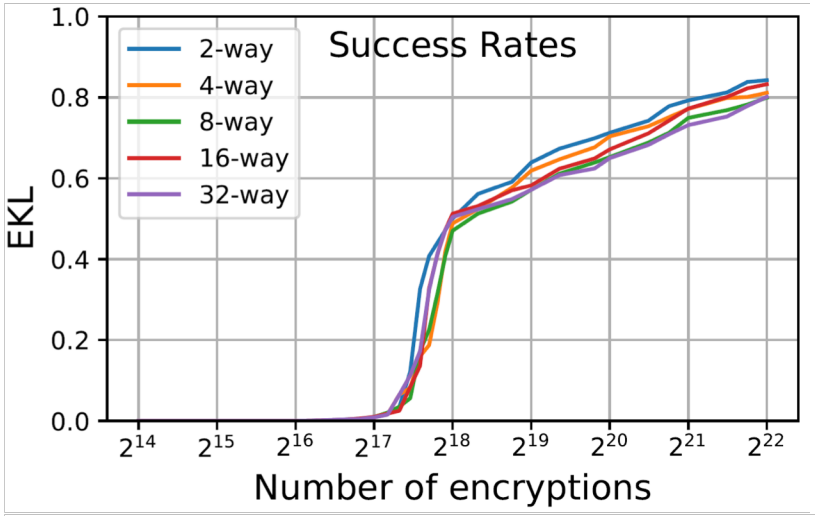
Reason: private cache's cache-misses dominate the AES computation time

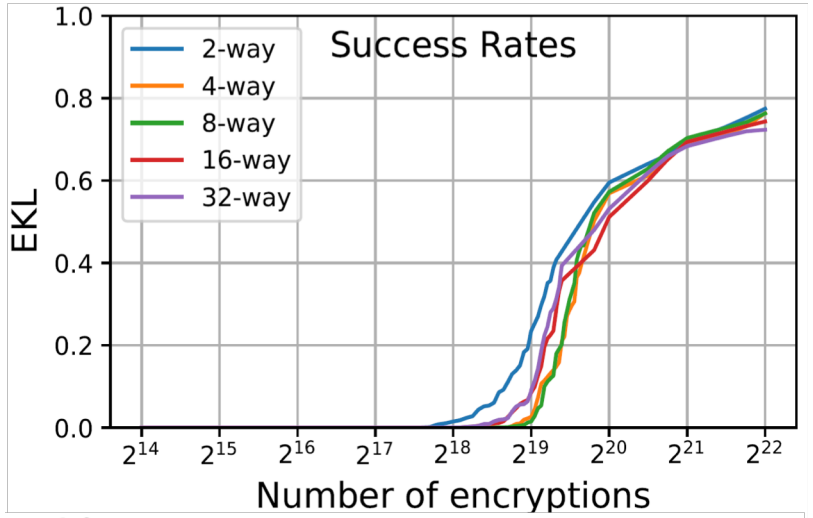
Measurement Results: SCS and SCA w/ and w/o Neighbor Processes

w/ Neighbor Processes

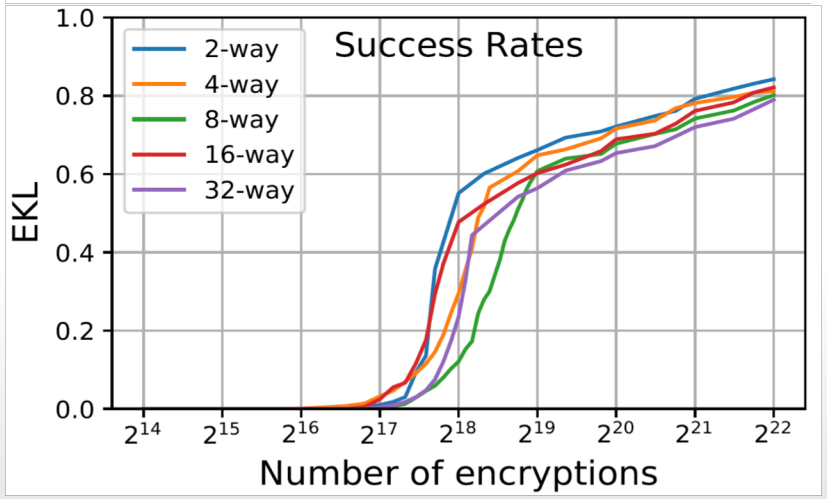
w/o Neighbor Processes

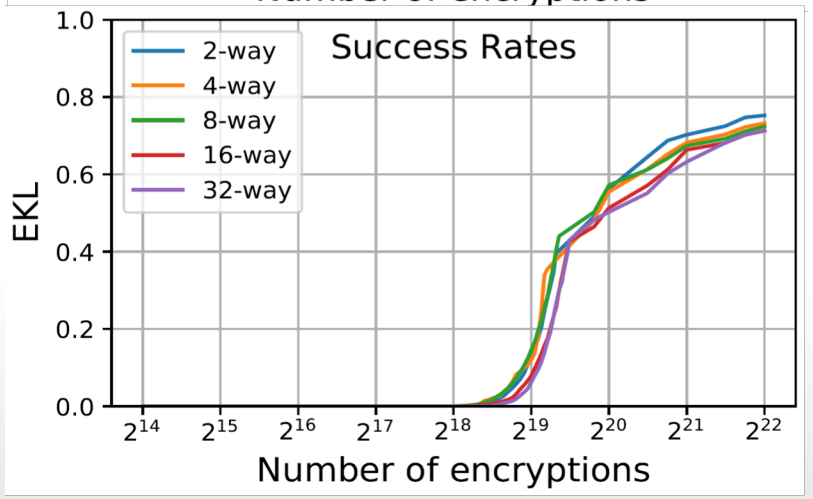
Shared Cache Size (SCS)





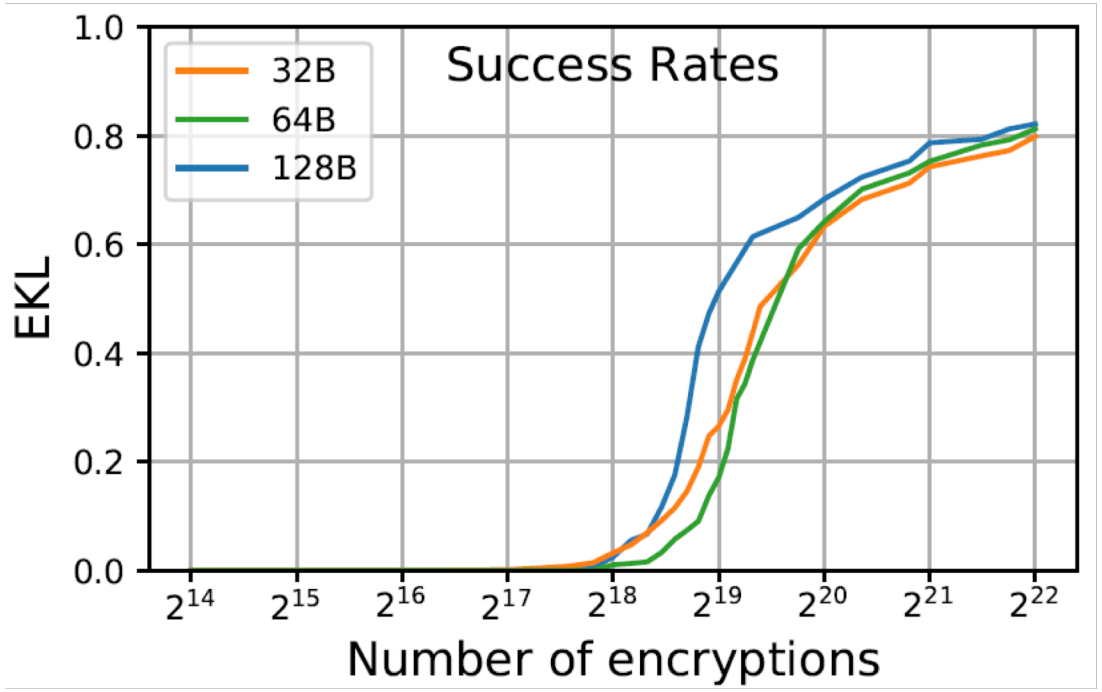
Shared Cache Associativity (SCA)





Measurement Results: CLS and RP

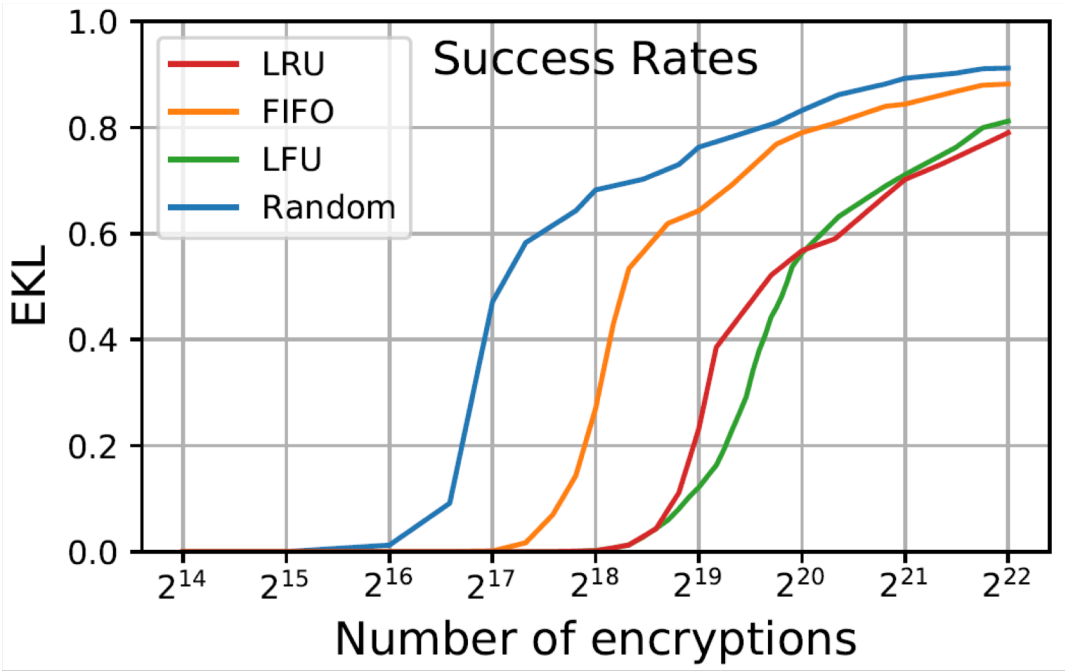
Cacheline Size (CLS)



Fact: insignificant impact

Reason: AES computation has good spatial locality

Replacement Policy (RP)



Fact: Random policy results in easiest attacks

Reason: AES computation has good spatial locality, but Random leverages no locality

Suggestions to the Attackers, Defenders, and System Designers

Takeaways:

a) Private cache configuration is the key
b) Shared cache configuration is trivial; adding the neighbor processes can increase the success rates;
c) Replacement policies and clusivity also can influence the attacks' success rates.

To attackers:

Binding a noise process with the same CPU of encryptions → easier attacks

To defenders:

a) Setting the private cache at the inflection points → optimal cost-efficiency balance
b) Using lock-into-cache instruction → more difficult attacks

To system designers:

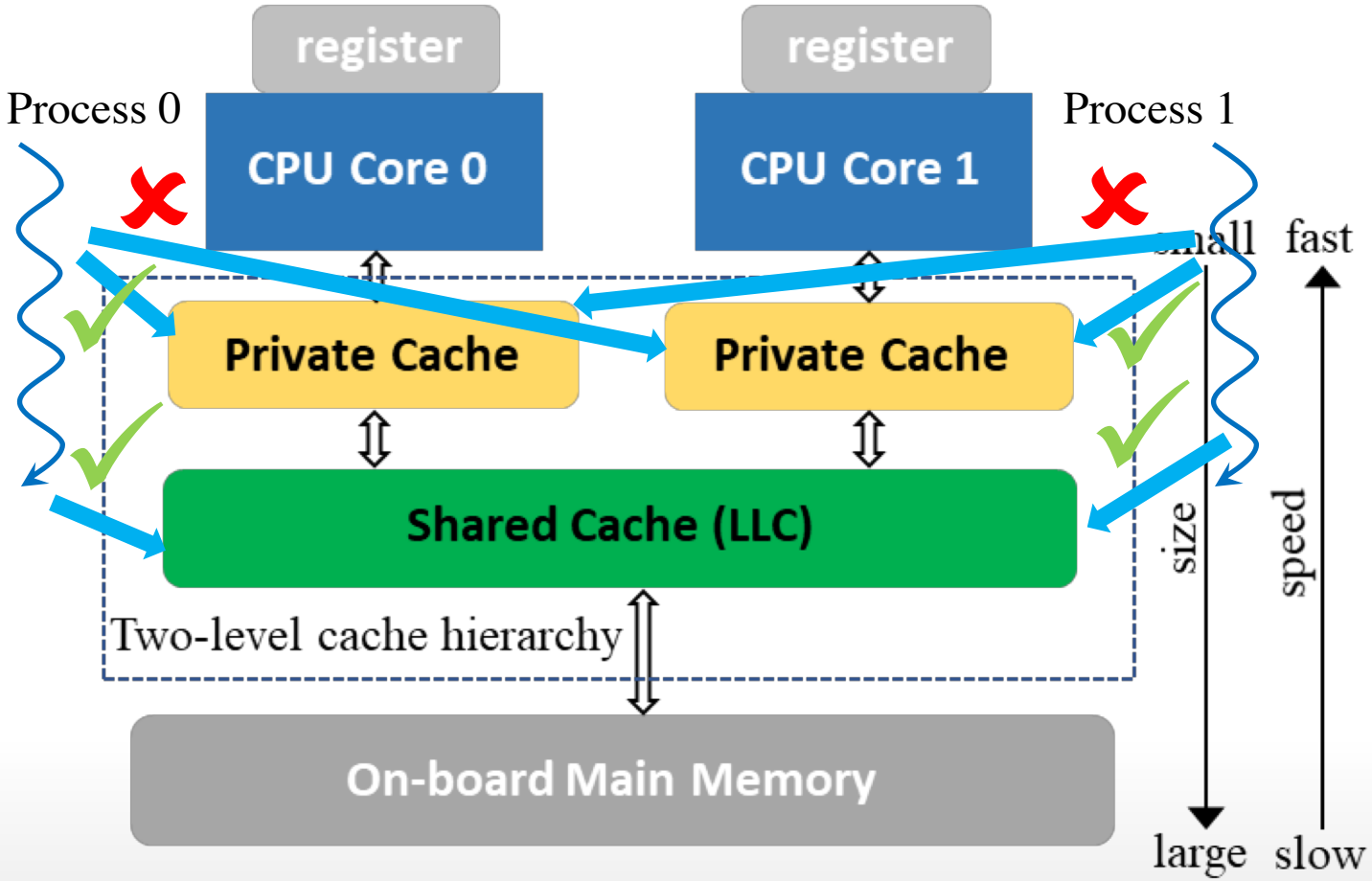
Heterogenous replacement policy and clusivity → good balance between system performance and security

Summary of Our Comparative Measurement Work

- We made the cache attack performances comparable
- We use the GEM5 platform to emulate the configurable caches
- We systematically each cache parameter's influences: the private cache is the key; the shared cache's impacts are trivial; The replacement policies and cache clusivity also have impacts

Backup Slides

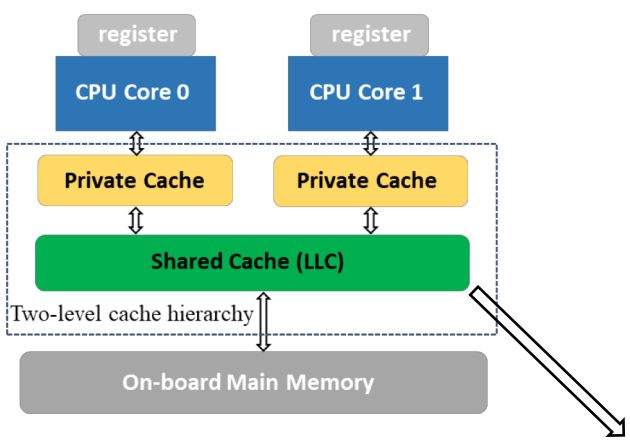
CPU Cache Model is a Two-Level Hierarchy

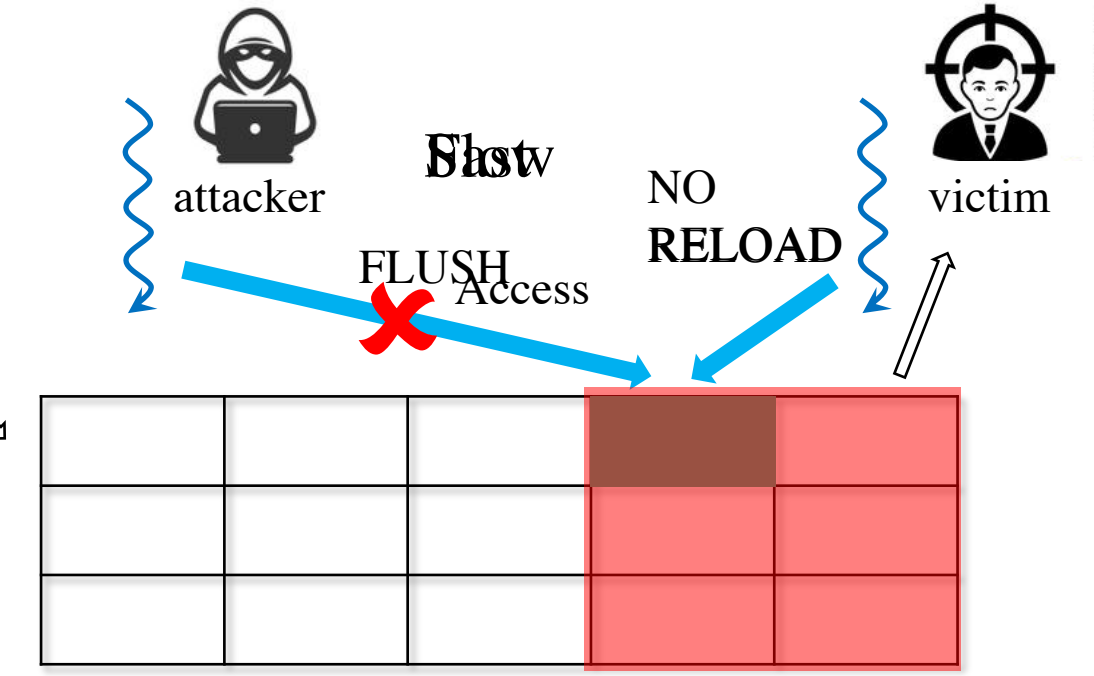


Two types of cache attacks:

- Access-driven attacks
- Time-driven attacks

Access-driven Attacks are Lightweight but Require Access Privileges





FLUSH+RELOAD Attack

The access-driven attacks are popular:

- It is more accurate
- It requires less computations

They require access privileges

- Revoking the privilege using Intel's Cache Allocation Technology (CAT) can prevent the attacks [Liu HPCA'16]

Conducting Comparative Measurements

Every GEM5 instances have the same system settings but different cache configurations

Execute the Bernstein's attack on AES on each GEM5 instances

Use the OpenSSL's AES implementation; OpenSSL precomputes the results of each AES step and stores them as a 4KB lookup table

Host System	
CPU	Intel(R) Xeon(R) E5-2620
main memory	192 GB RDIMM
GEM5 Platform	
CPU core #	2 cores (3 GHz)
main memory	4 GB
CPU cache	two-level configurable
operating system	Ubuntu 16.04.1 LTS
OpenSSL version	1.0.2 LTS

Limitations and Future Research Directions

Limitations:

a) Unclear whether the measurement is compatible with other cache side-channel attacks
b) The measurement does not count the effects of some modern hardware technologies

Future directions:

a) Study whether this measurement's approach, findings, and conclusions transferable to other cache side-channel attacks
b) Study whether the RISC-based embedded systems' cache configurations have the same or similar impacts
c) Study how accurate this emulation-based measurement is; try to exploit the findings to build a prediction model for the cache timing attack vulnerability of unseen systems