Deconstructing New Cache Designs for Thwarting Software Cache-based Side Channel Attacks

Jingfei Kong*	University of Central Florida
Onur Acıiçmez	Samsung Electronics
Jean-Pierre Seifert	Samsung Electronics
Huiyang Zhou	University of Central Florida

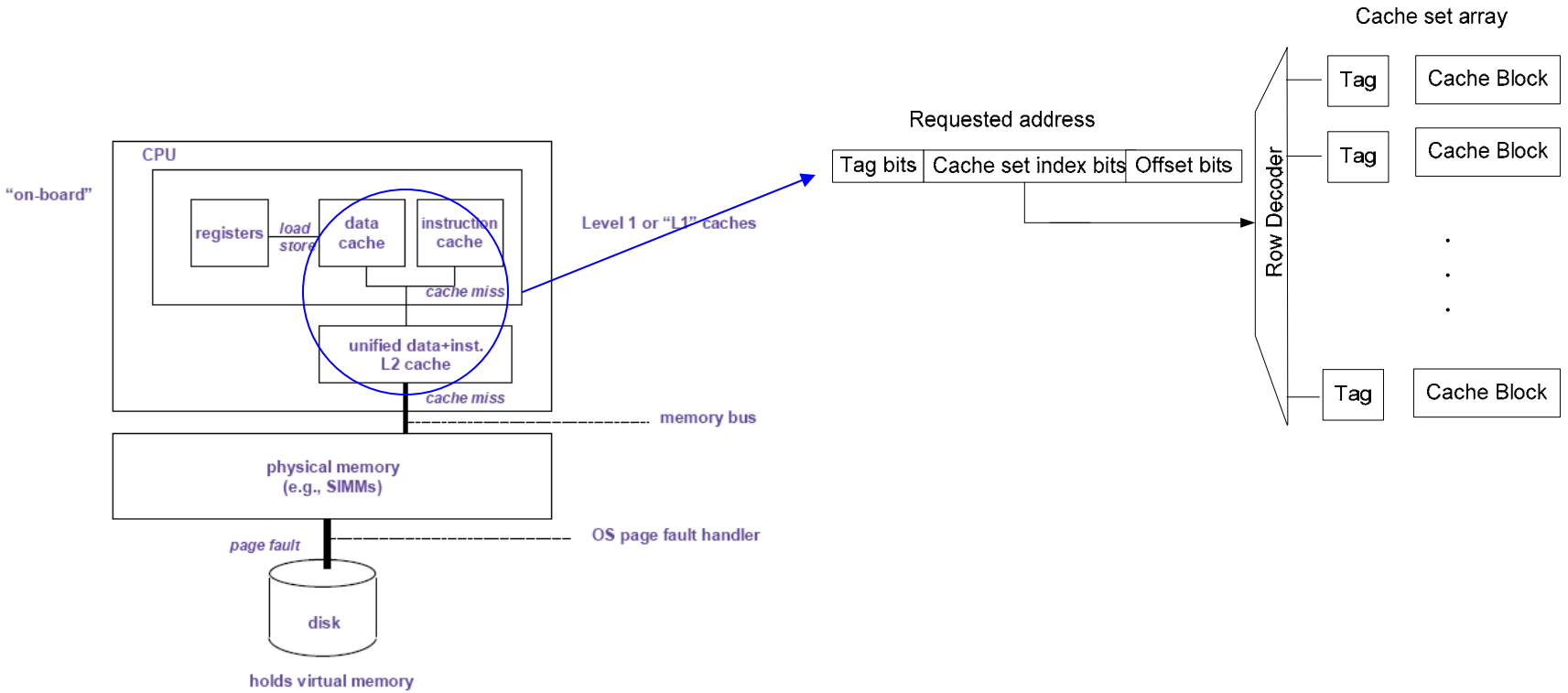
Outline

- Background (more details are in related papers)
- Security evaluation on previously proposed secure cache designs
- Potential solutions and conclusion

What are software cache-based side channel attacks?

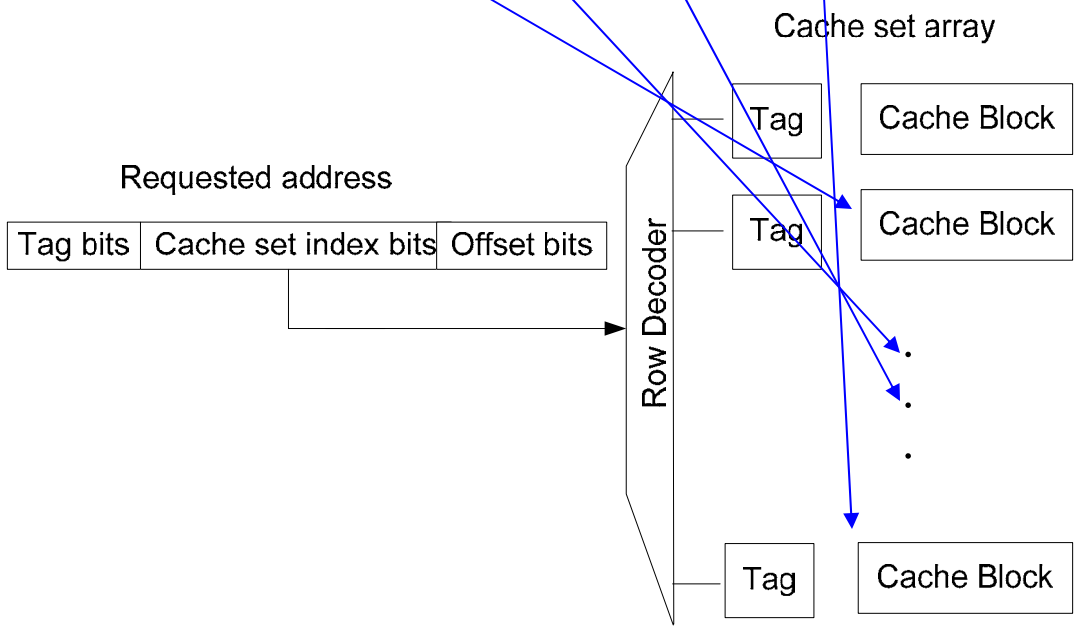
- Side channel attacks
  - Exploit any observable information generated as a byproduct of the cryptosystem implementation
  - Infer the secret information
- Cache implementation of modern general-purpose microprocessors
  - A hardware component sits between the CPU pipeline and main memory
  - Take advantage of spatial and temporal locality to minimize communication latency
- Software-managed observation
  - No need for physical possession of the attacked system

Cache design of modern general-purpose microprocessors



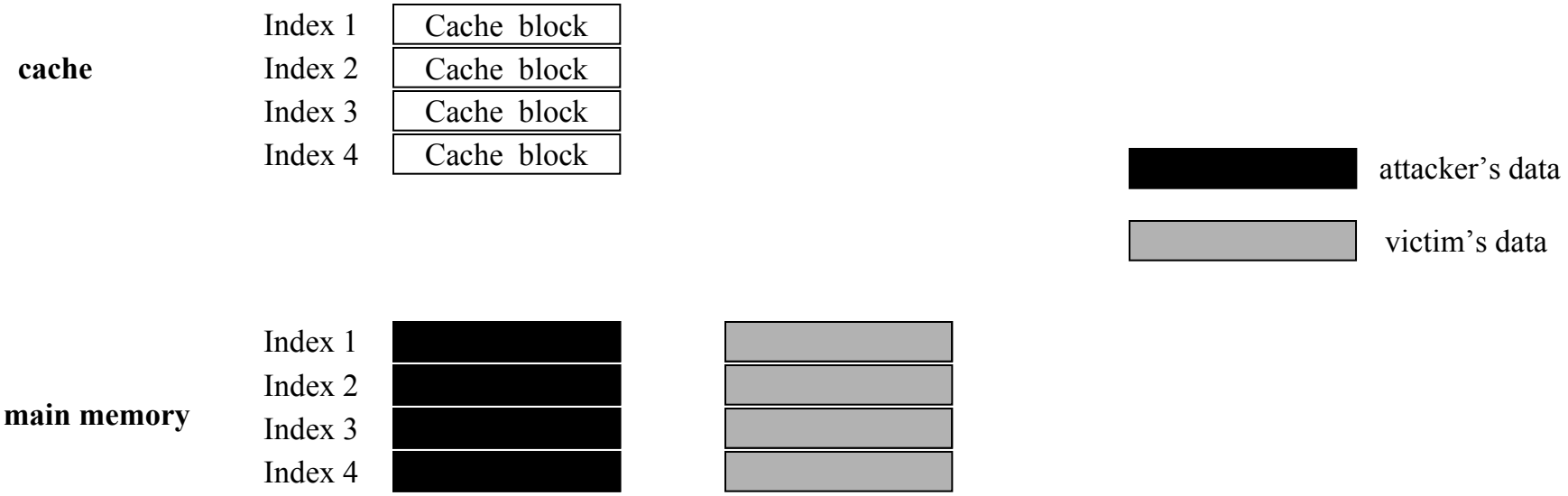
The case of Advanced Encryption Standard (AES)

$$\begin{Bmatrix} x_0^{i+1}, x_1^{i+1}, x_2^{i+1}, x_3^{i+1} \\ x_4^{i+1}, x_5^{i+1}, x_6^{i+1}, x_7^{i+1} \\ x_8^{i+1}, x_9^{i+1}, x_{10}^{i+1}, x_{11}^{i+1} \\ x_{12}^{i+1}, x_{13}^{i+1}, x_{14}^{i+1}, x_{15}^{i+1} \end{Bmatrix} = \begin{Bmatrix} T_0[x_0^i] \oplus T_1[x_5^i] \oplus T_2[x_{10}^i] \oplus T_3[x_{15}^i] \oplus \{k_0^i, k_1^i, k_2^i, k_3^i\} \\ T_0[x_4^i] \oplus T_1[x_9^i] \oplus T_2[x_{14}^i] \oplus T_3[x_3^i] \oplus \{k_4^i, k_5^i, k_6^i, k_7^i\} \\ T_0[x_8^i] \oplus T_1[x_{13}^i] \oplus T_2[x_2^i] \oplus T_3[x_7^i] \oplus \{k_8^i, k_9^i, k_{10}^i, k_{11}^i\} \\ T_0[x_{12}^i] \oplus T_1[x_1^i] \oplus T_2[x_6^i] \oplus T_3[x_{11}^i] \oplus \{k_{12}^i, k_{13}^i, k_{14}^i, k_{15}^i\} \end{Bmatrix}$$



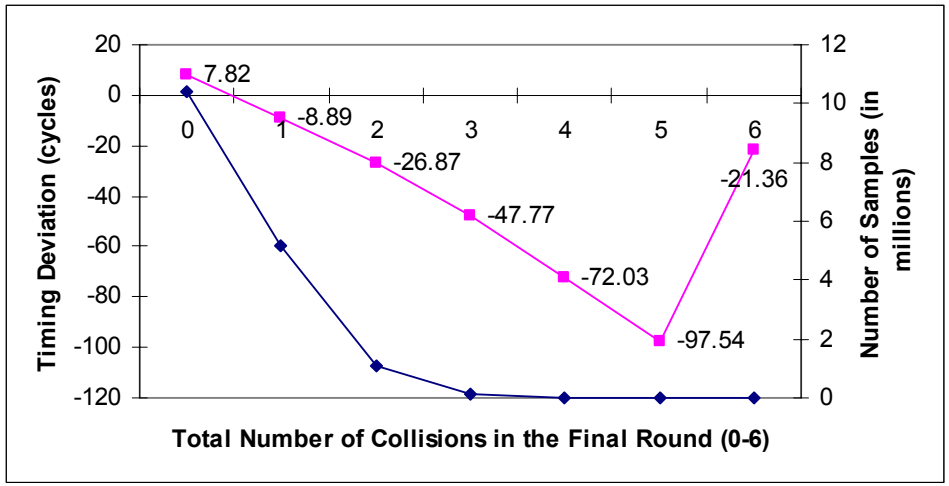
Access-driven attacks

- The attacker does not have direct access to the victim process' memory address space
- The attacker controls one or multiple process(es) which share the cache with the victim process



Time-driven attacks

- The attacker sends the encryption request to the victim process and upon response records the execution time



Why hardware support against software cache-based attacks?

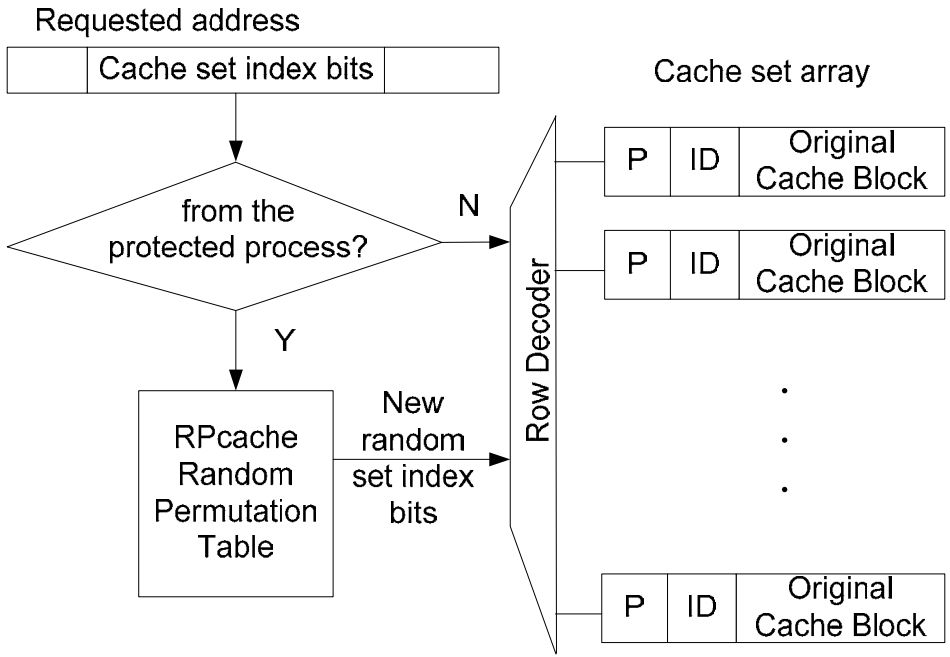
- Generic
- Secure against both attacks
- Performance efficient

Partition Locked Cache (PLcache) & Random Permutation Cache (RPcache)

PLcache

- Control interface
  - ld.lock/ld.unlock and st.lock/st.unlock
  - lock_mem_region(addr, length) unlock_mem_region(addr, length)

RPcache



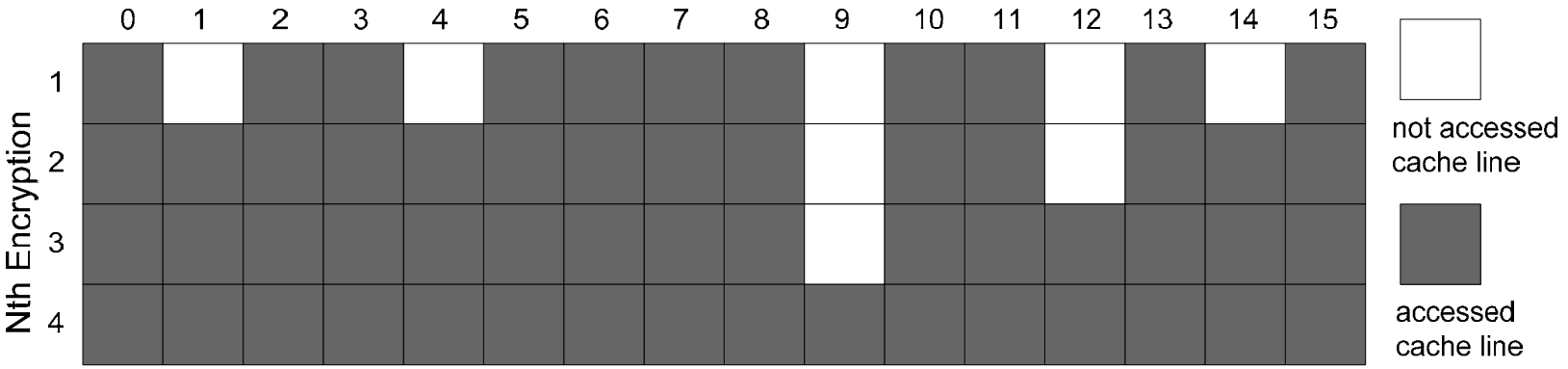
Previous work from Wang et.al at International Symposium on Computer Architecture (ISCA) 2007

Simulation Environment

Superscalar Core	7-stage pipeline: Fetch/Dispatch/Issue/RegisterRead /EXE/WriteBack/Retire
	Fetch/Dispatch/Issue/ MEM issue/Retire Bandwidth: 4
	Fully-symmetric Function Units: 4
	Reorder Buffer size: 64
	Issue Queue Size: 32
	Load Store Queue Size: 32
Execution Latencies	Address Generation: 1 cycle
	Memory Access: 2 cycles (hit in data cache)
	Integer ALU ops: 1 cycle
	Complex ops:MIPS R10000 latencies
Instruction Cache	32KB 2-way, Block size: 64B 10-cycle miss penalty
L1 Data Cache	32KB 2-way, Block size: 64B 10-cycle miss penalty
L2 Unified Cache	2MB 16-way, Block size: 64B 300-cycle miss penalty

Security Evaluation of PLcache

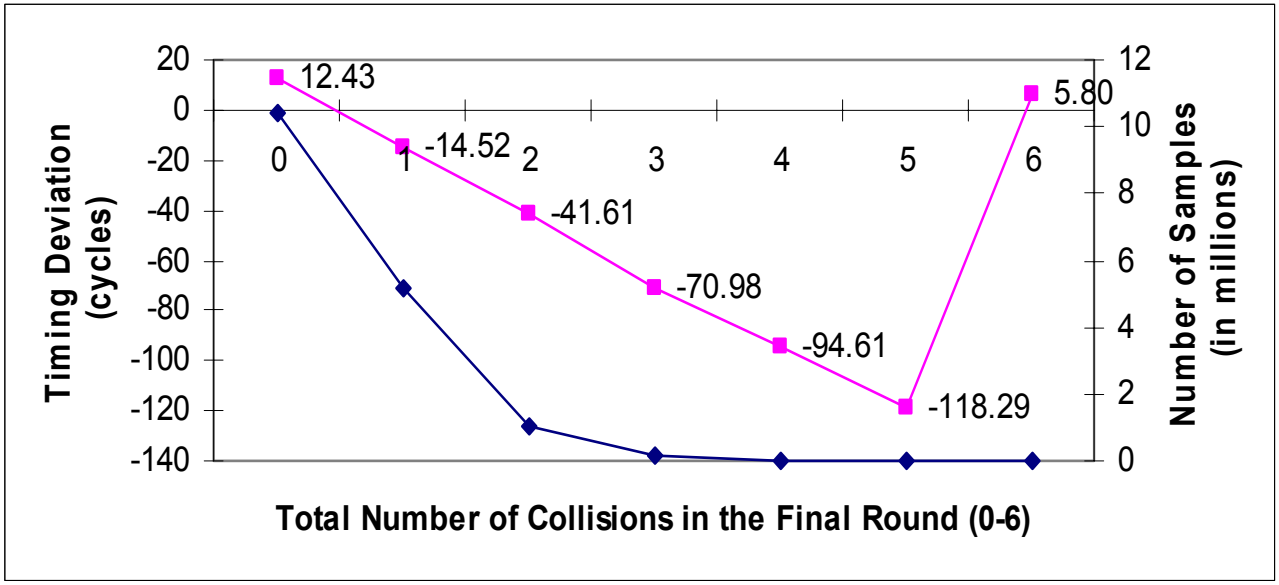
- Initial loading exposure of PLcache



- Potential vulnerability to denial of service attacks

Security Evaluation of RPcache

- Vulnerable to latest cache-collision timing attacks



Possible solutions and future work

- Secure PLcache
  - Using locking mechanism to do the preloading first before cryptographic operations take place
  - Allow the locked cache lines to be used by other processes when the owner process is inactive
- Secure RPcache
  - Preloading only supported from OS is not enough
  - The best possible solution is to detect the event of cache misses in hardware and respond to the event in software

Summary

- Latest software cache-based side channel attacks are emerging threats
- Current secure cache designs, although providing generic security protection and incurring small performance overhead, are still missing some important security issues

Thank you!

Questions?