可抵御缓存侧信道攻击的随机化缓存设计

宋威，薛子涵

中国科学院信息工程研究所
网络空间安全防御重点实验室

RISC-V 中国峰会　2024年8月23日

主要内容

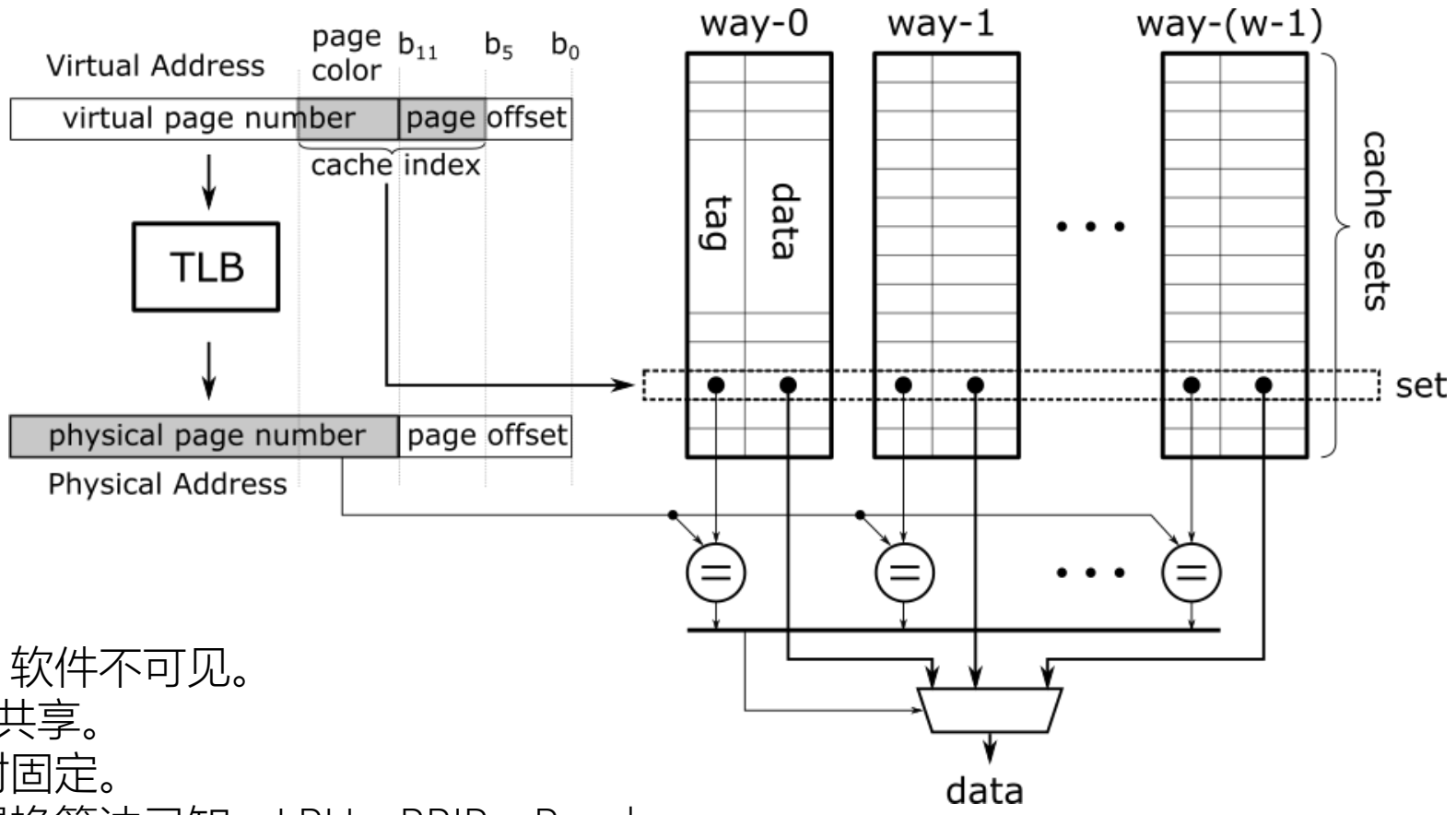
缓存侧信道攻击是攻击者通过在受害机器上运行恶意程序（非接触式）从缓存中获取关键信息的一种信息泄露的攻击方式。近年来被大量采用。

缓存随机化是近年来提出的一种纯硬件的缓存防御方法，可以抵御冲突型的缓存侧信道攻击。

现在还没有缓存随机化防御在真实处理器上的实现。那么，我们来做一个！

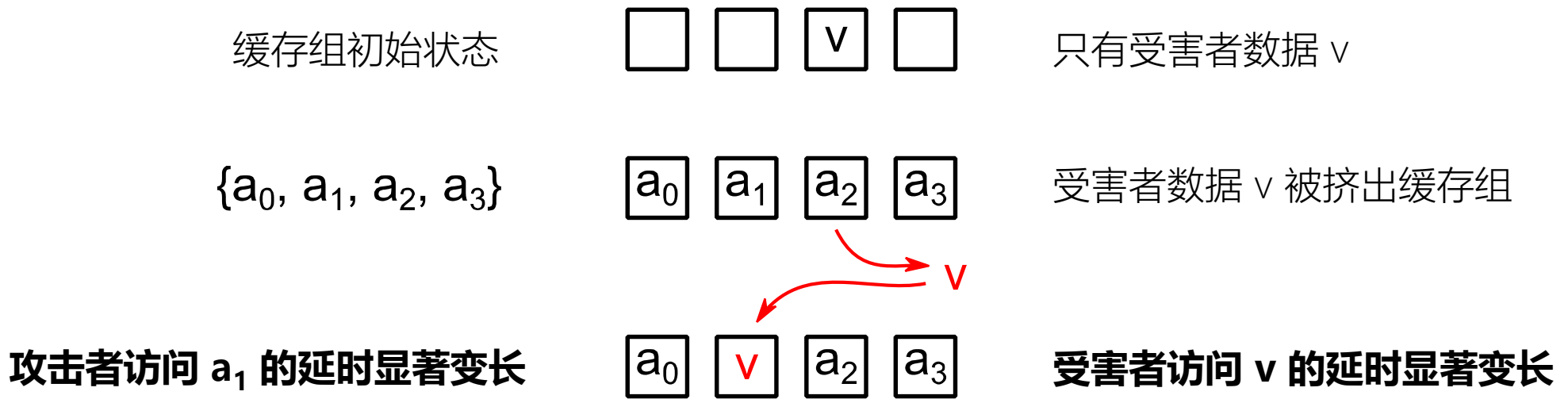
- 冲突型缓存侧信道攻击
- 缓存随机化防御
- 我们的优化
- 在Rocket-Chip上的实现

缓存的基本结构



- 缓存是微体系结构模块，软件不可见。
- 末级缓存（LLC）被多核共享。
- 内存地址到缓存组的映射固定。
- 缓存冲突时触发置换，置换算法已知：LRU，RRIP，Random。

冲突型缓存侧信道攻击



- 正常情况下，受害者的目标数据 v 在缓存命中
- 攻击者通过访问一个驱逐集 $\{a_0, a_1, a_2, a_3\}$，强迫将 v 挤出缓存.
- 受害者再次访问 v 的延时显著变长，攻击者访问驱逐集的时间也变长。

$\{a_0, a_1, a_2, a_3\}$ 和 v 在缓存中保存在同一个缓存组 (congruent)

（最小）驱逐集：一个包含足够多congruent地址的集合，访问它可以将目标地址 v 驱逐出缓存。

驱逐集寻找算法

在早期的电脑上，驱逐集是算出来的。在现在的电脑上，驱逐集是找出来的。（原因：虚拟地址到物理地址映射被保护，Intel的复杂末级缓存分片寻址方式）

- 组消除[1,**2**,3]（GE: Group Elimination）：80ms, 80%
- 覆盖-裁剪-触发算法[3,4]（PPP: Prime, Prune and Probe）：1.4ms, 24%
- 冲突测试[3]（CT: Conflict Testing）：18ms, 70%

其他：W+W[5]（10ms, 5%），CTPP[**6**]（1.3ms, 92%）

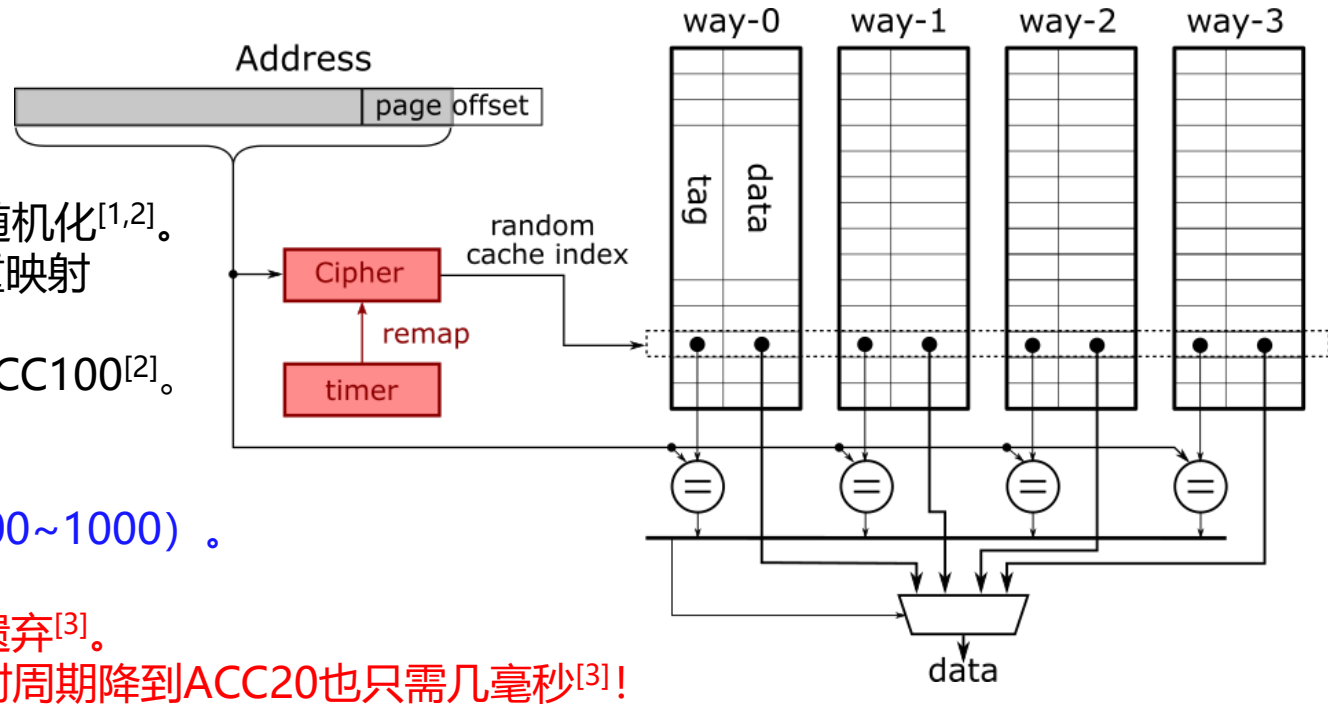
[1] P. Vila, B. Köpf, J. F. Morales. "Theory and practice of finding eviction sets." S&P 2019.
[2] W. Song, P. Liu. "Dynamically finding minimal eviction sets can be quicker than you think for side-channel attacks against the LLC." RAID 2019.
[3] M. K. Qureshi. "New attacks and defense for encrypted-address cache." ISCA 2019.
[4] A. Purnal, I. Verbauwhede. "Advanced profiling for probabilistic Prime+Probe attacks and covert channels in ScatterCache." ArXiv, Aug 2019.
[5] J. P. Thoma, T. Güneysu. "Write me and I'll tell you secrets – Write-after-write effects on Intel CPUs." RAID 2022.
[6] Z. Xue, J. Han, W. Song. "CTPP: A fast and stealth algorithm for searching eviction sets on Intel processors." RAID 2023.

随机化的多路组相联缓存

- 地址到缓存组的映射由加密模块（Cipher）随机化[1,2]。
- 周期性修改加密密钥，更新缓存映射，称为重映射（remap）[2]。
- 重映射的周期为一定量的缓存访问，比如说ACC100[2]。

- 页内偏移不能被利用（时间代价 x64）。
- 寻找攻击目标依靠缓存组扫描（时间代价 x100~1000）。

- 每次重映射会造成40%到60%的缓存数据被遗弃[3]。
- 冲突测试算法能在ACC30找到驱逐集，重映射周期降到ACC20也只需几毫秒[3]！

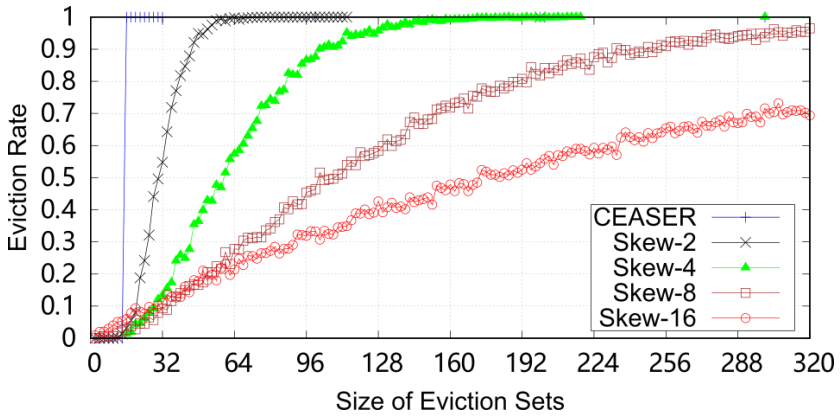


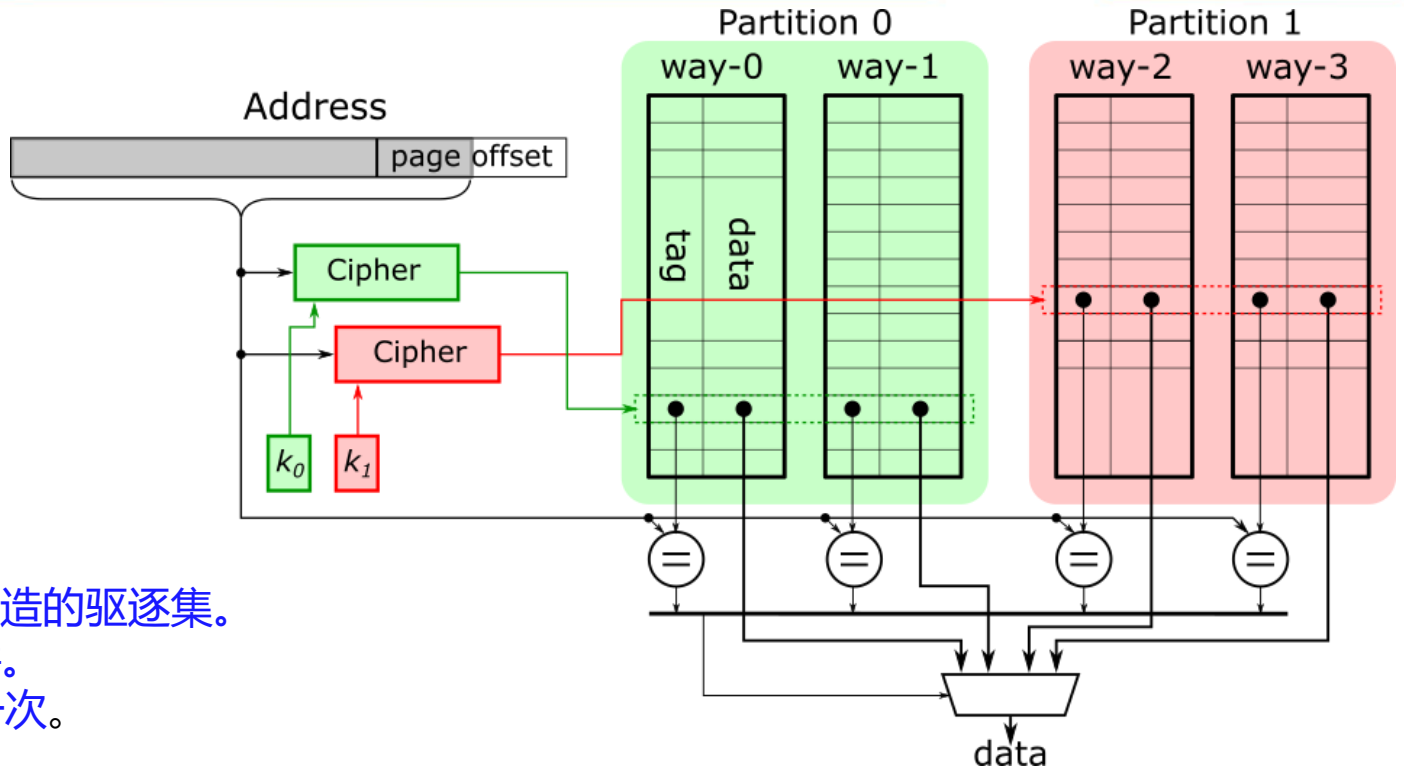
[1] W. Song, R. Hou, D. Meng. “Defeating the recent AnC attack by simply hashing the cache indexes — implemented in a BOOM SoC”. RISC-V Workshop 2018.

[2] M. K. Qureshi. “CEASER: Mitigating conflict-based cache attacks via encrypted-address and remapping”. MICRO 2018.

[3] W. Song, B. Li, Z. Xue, et al. “Randomized last level caches are still vulnerable to cache side channel attacks! But we can fix it.” S&P 2021.

动态随机化Skewed缓存





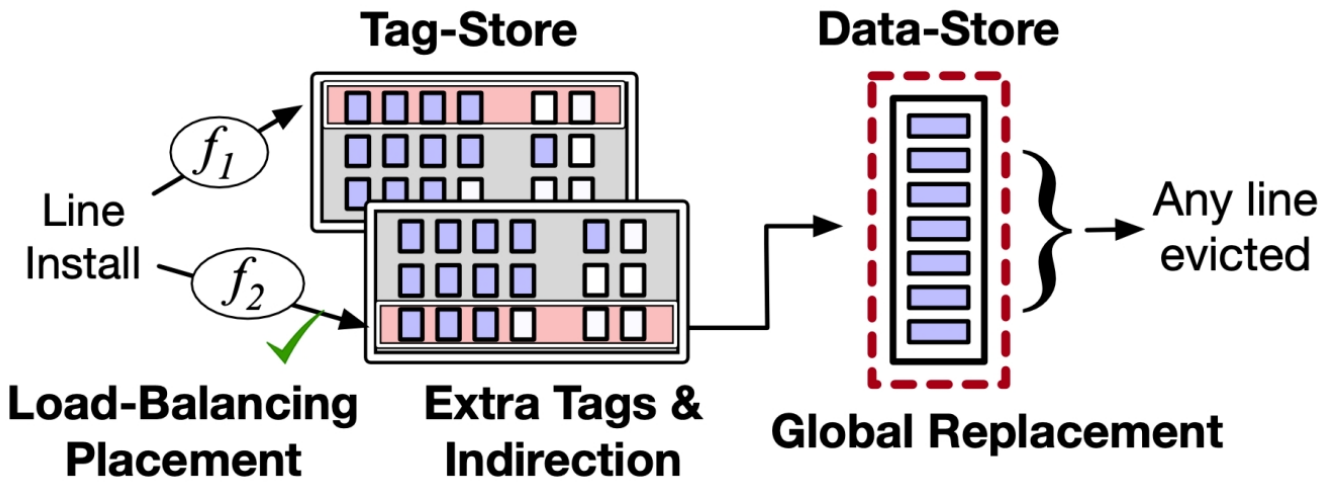
- 引入Skewed缓存[1]

- 攻击者只能构造由部分congruent地址构造的驱逐集。
- 使用驱逐集挤出目标地址变成了概率事件。
- 如果没有clflush指令，驱逐集只能使用一次。

- 即使使用成功极低的驱逐集，利用重复攻击仍然可以泄露信息[2]!
- 在ACC100内，攻击者仍然可以构造成功率高于70%的驱逐集[3]!

[1] M. K. Qureshi. "New attacks and defense for encrypted-address cache." ISCA 2019.
[2] T. Bourgeat, J. Drean, Y. Yang, et al. "CaSA: End-to-end quantitative security analysis of randomly mapped caches." MICRO 2020.
[3] W. Song, B. Li, Z. Xue, et al. "Randomized last level caches are still vulnerable to cache side channel attacks! But we can fix it." S&P 2021.

MIRAGE



G. Saileshwar, M. Qureshi. “MIRAGE: Mitigating conflict-based cache attacks with a practical fully-associative design.” Security 2021.

- **目标：彻底消除攻击者控制的缓存驱逐事件。**

- Skewed缓存：缓存随机化和多种缓存映射，提供原始随机分布来源。
- 元数据冗余供给+skew平衡分布：极度降低由元数据存储造成的缓存冲突。
- 元数据和数据存储分离：降低数据存储空间，并实现全局随机置换。

- 攻击者构造一次受控的缓存冲突需要上万年？！
- 22%的SRAM存储空间代价！（模型估计的运行时性能代价3%？）

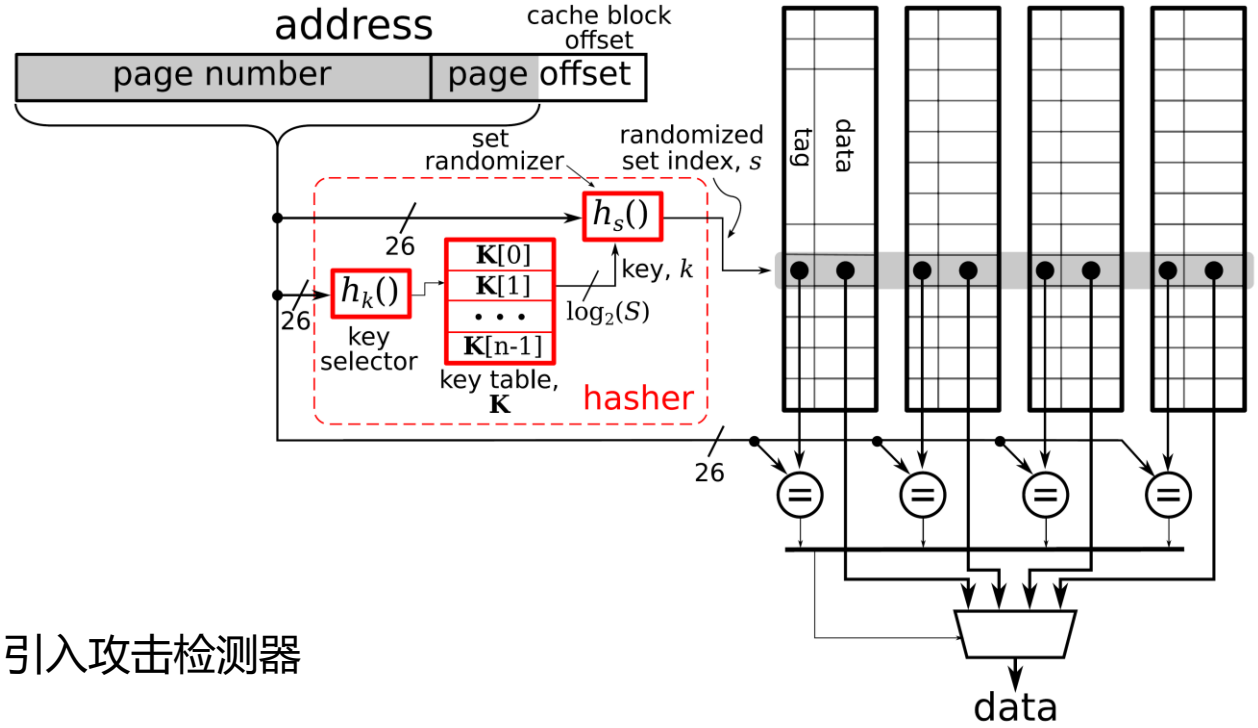
我们真的需要这样的缓存架构吗？

我们的构想：**随机化传统多路组相联缓存**

影响性能的安全方案往往不会被接受，只有性能影响小，可在现有架构上打补丁的安全方案才容易被现有处理器采用！

- 问题：
  4-7周期的随机映射延时
  重映射的代价高（40~60%数据丢失）
  重映射过于频繁（ACC10）

- 我们的解决方案：
  - 单周期的随机映射算法
  - 引入多步置换链，降低数据丢失量至10%
  - 将重映射周期的计数单位改成缓存驱逐事件，引入攻击检测器



- 基于Rocket-Chip的FPGA原型实现：发表于IEEE Transactions on Computers 2024, vol 73, no. 4

https://github.com/comparch-security/chipyard-random-llc

2.84% 面积 (1.6% SRAM)，小于0.9% CPI, 2.1% 功耗　（多核OoO下会更低）

单周期的随机映射

现有设计

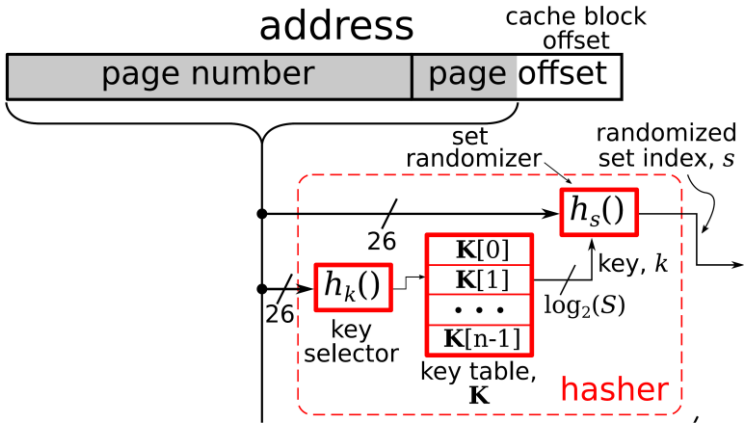
CEASER 和CEASER-S：**2周期**
4级Feistel-Network (LLBC)
线性算法，被证明强度不够。

ScatterCache: **5周期** 5级QARMA-64

MIRAGE: **3周期** PRINCE

根据我们在Rocket-Chip上的实测，随机映射每增加1个周期，CPI损失0.4~0.8%。

结果：随机映射直接造成1~4%的CPI损失。

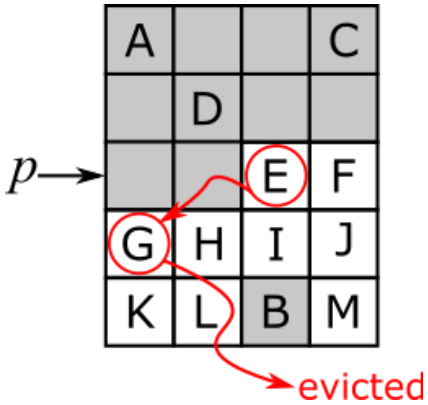


我们的设计

- 用一个单周期非线性哈希函数代替加密算法。
- 引入一个随机种子池，不同地址选择不同的种子，避免碰撞攻击。
- 随机算法的破解难度大于驱逐集构造难度。
- 重映射时更新种子池。

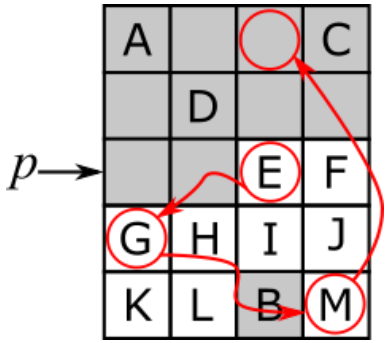
结果：单周期映射，难度足够，免于攻击。

利用多步置换链降低数据丢失



传统重映射：
按顺序对所有缓存块进行移位。
当新的缓存组没有空闲位置时，按照置换算法选择一个缓存块（G），将其驱逐。

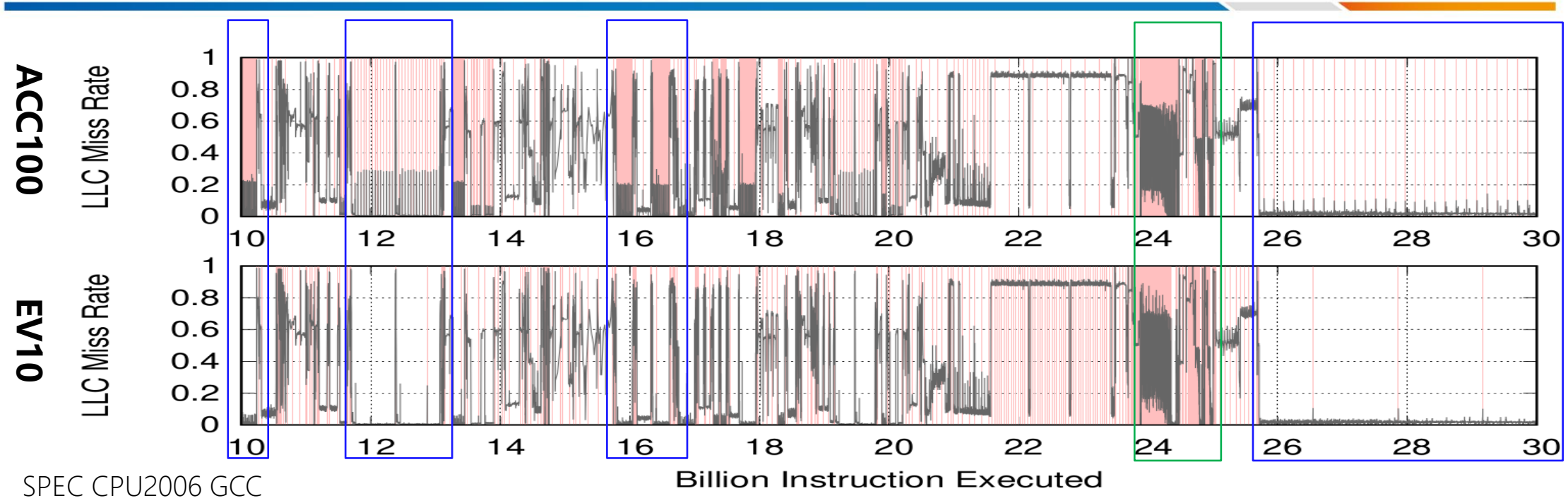
结果：50~60%的缓存块在重映射过程中丢失。



多步置换链（贪心算法）：
当新的缓存组没有空闲位置时，按照置换算法选择一个缓存块（G），将缓存块（E）存入。
继续将被置换出的缓存块（G）作为移位目标，继续移位，直至找到一个空闲位置，或者目标缓存组的所有缓存块都已经被重映射。

结果：~10%的缓存块在重映射过程中丢失。

替换重映射周期计数单位：用EV10代替ACC100



SPEC CPU2006 GCC

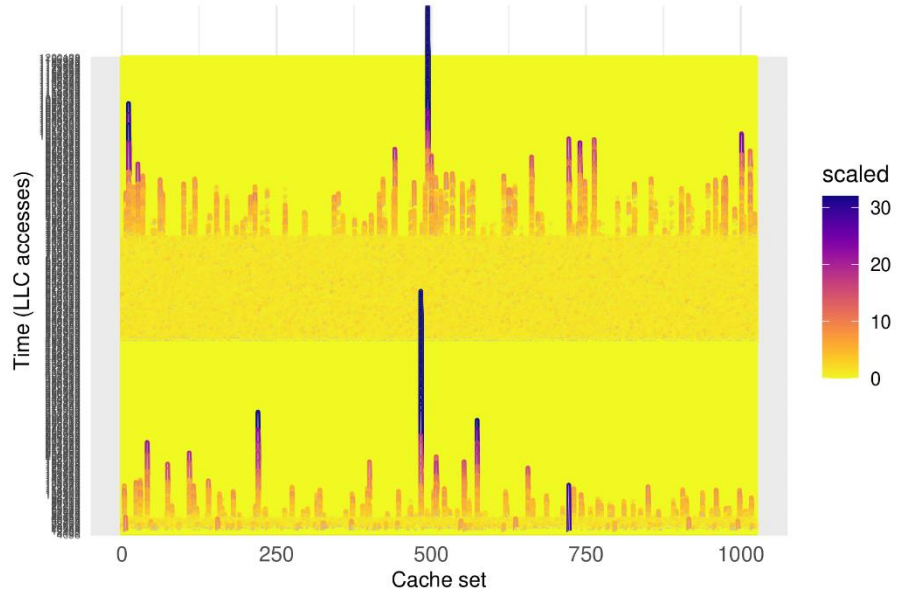
- 由于缓存的过滤特性，当缓存足够预热，在攻击时末级缓存的**访问数**和**驱逐数**是**相等**的！
- 对于正常程序来说，缓存性能好意味着命中率高，**缓存驱逐率极低**！
- 当缓存驱逐率高时，ACC和EV计数都会造成高的重映射频率，但是**缓存本身的性能已经很低**！
- 如果缓存的随机映射导致了更多的冲突，这些冲突引发重映射可能会导致一个更好的映射，提高缓存性能！

攻击检测器：GE和PPP算法

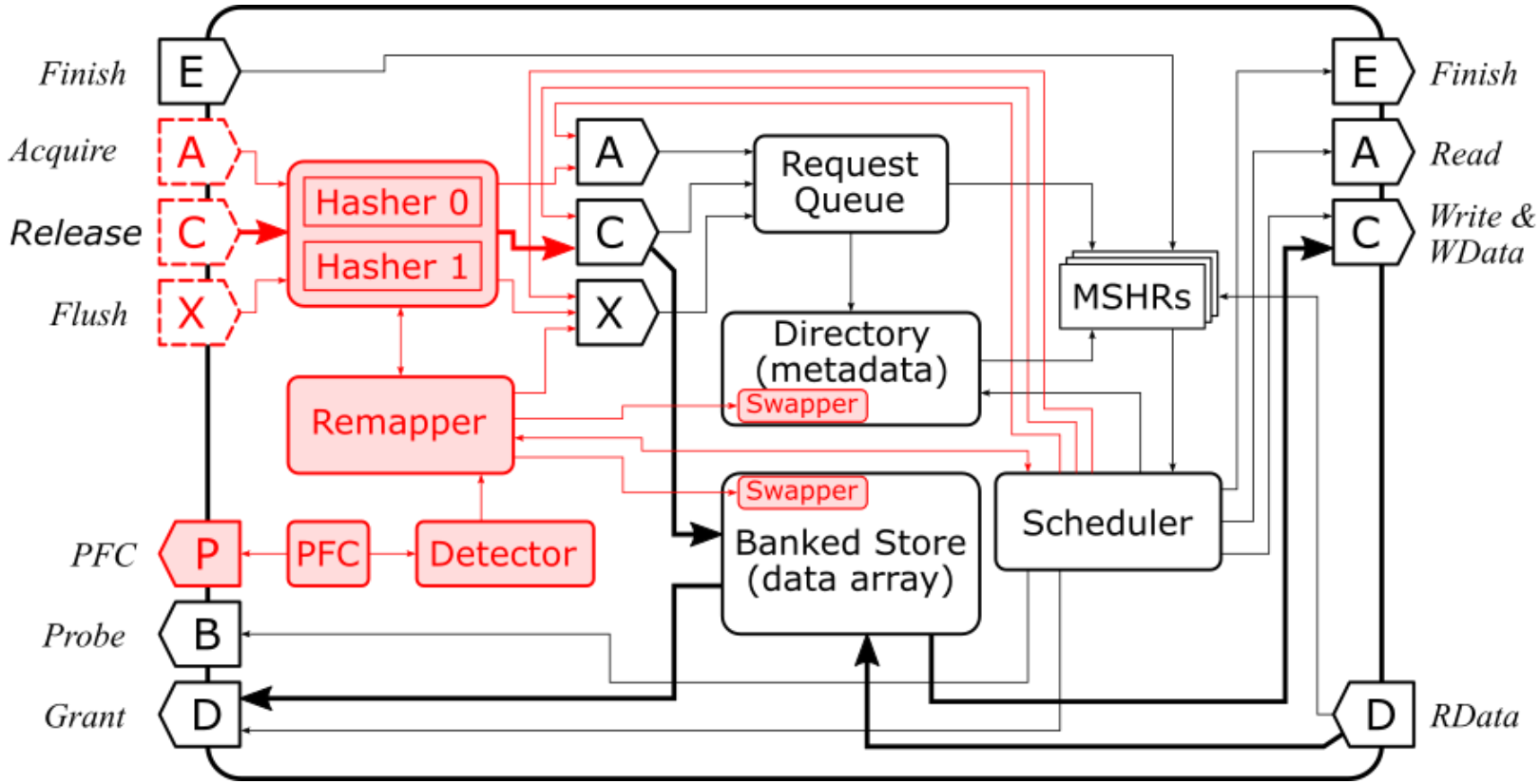
对于GE和PPP这些将大驱逐集裁剪成小驱逐集的算法，我们发现算法会在目标缓存组上造成较高的缓存驱逐事件，导致缓存驱逐事件在缓存组上的不均匀分布。

使用Z-Score对缓存组上的置换事件做分布标准化。可以较为准确的检测GE和PPP算法。

利用Z-Score在线检测器实现缓存的按需动态重映射。进一步降低固定重映射周期。



在Rocket-Chip上的实现



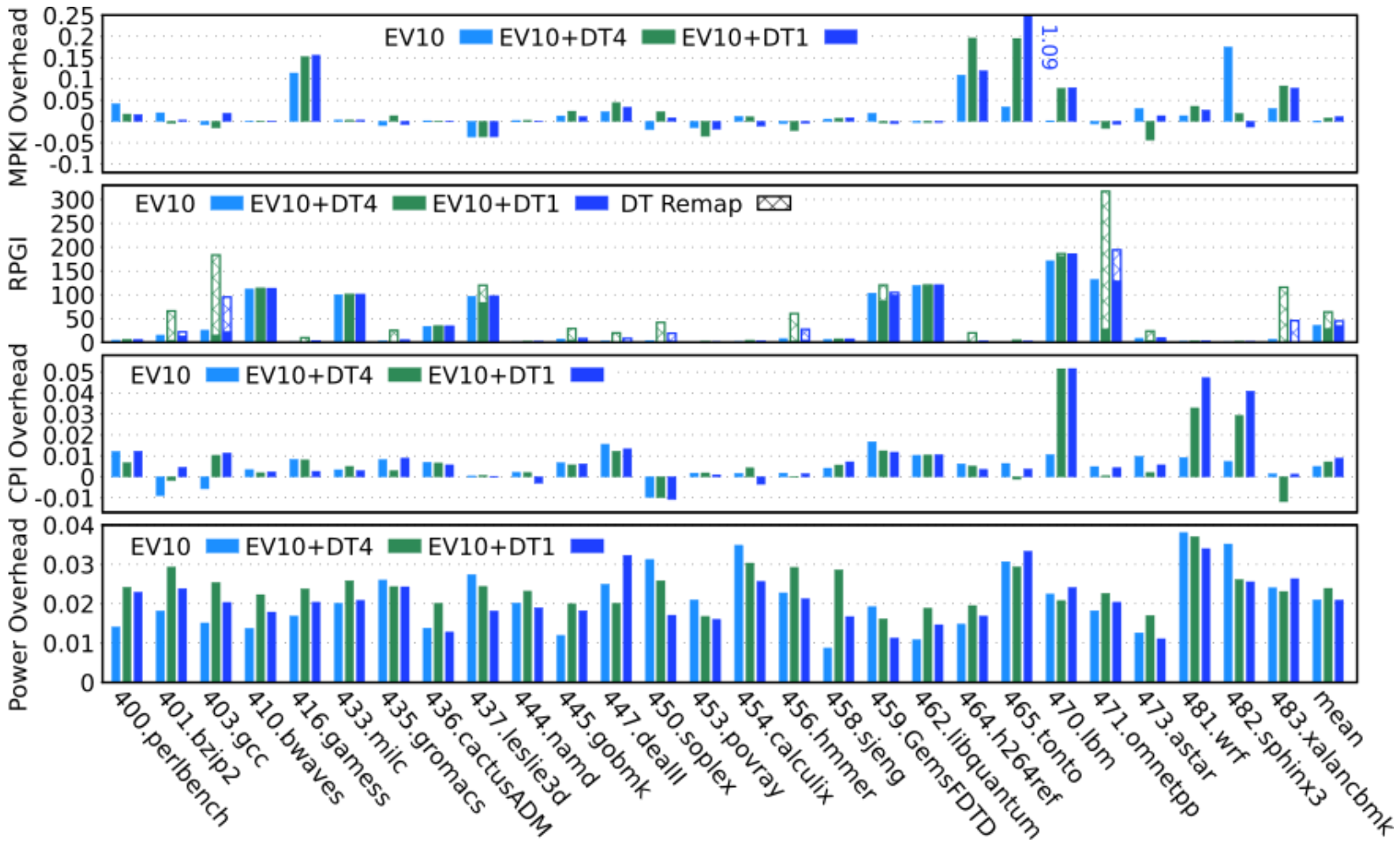
防御效果

	Detector Combination					
Alg.	**Static**	**EV10**	**DT4**	**DT1**	**EV10+DT4**	**EV10+DT1**
GE	97.2%	~0.0%	~0.0%	~0.0%	~0.0%	~0.0%
PPP	20.4%	13.3%	~0.0%	~0.0%	~0.0%	~0.0%
CT	14.8%	~0.0%	~0.0%	~0.0%	~0.0%	~0.0%
CT-fast	13.9%	2.5%	1.0%	0.1%	0.3%	~0.0%
W+W	17.4%	11.7%	~0.0%	~0.0%	~0.0%	~0.0%

面积代价 （ASIC估计）

		Logic	RAM	Total	Percent	Overhead
Original	Rocket-Chip	755.9	4713	5469		
	LLC	106.2	4175	4282	78.3%	
	Channels	64.86	0	64.86	1.19%	
	MSHRs	8.287	0	8.287	0.15%	
	Request Queue	16.79	0	16.79	0.30%	
	Banked Store	7.544	3991	3999	73.1%	
	Directory	3.480	184.4	187.9	3.44%	
Random	Rocket-Chip	833.7	4790	5624		2.84%
	LLC	171.4	4253	4425	78.7%	3.34%
	Channels	68.39	0	68.39	1.22%	5.44%
	MSHRs	11.98	0	11.98	0.21%	44.60%
	Request Queue	22.74	0	22.74	0.40%	35.45%
	Banked Store	14.12	3991	4005	71.2%	0.16%
	Directory	6.154	240.9	247.0	4.39%	31.49%
	Hasher	16.71	0	16.71	0.30%	
	Detector	10.56	21.21	31.78	0.56%	
	Remapper	0.750	0	0.750	0.01%	
	PFC	9.528	0	9.528	0.17%	

性能代价


MPKI Overhead
EV10
EV10+DT4
EV10+DT1
1.09
RPGI
DT Remap
CPI Overhead
Power Overhead
400.perlbench
401.bzip2
403.gcc
410.bwaves
416.gamess
433.milc
435.gromacs
436.cactusADM
437.leslie3d
444.namd
445.gobmk
447.dealII
450.soplex
453.povray
454.calculix
456.hmmer
458.sjeng
459.GemsFDTD
462.libquantum
464.h264ref
465.tonto
470.lbm
471.omnetpp
473.astar
481.wrf
482.sphinx3
483.xalancbmk
mean

总结

- 缓存随机化可以用来抵御冲突型的缓存侧信道攻击
- 观点
  - 基于skewed缓存的随机化方案性能代价过高。
  - 完全消除攻击者控制的缓存冲突是没有必要的。
  - 传统的多路组相联缓存可以做到足够安全。
  - 该技术现在已经可以使用了。
- 我们的改进
  - 单周期的随机映射
  - 基于置换链的重随机
  - 使用缓存驱逐事件为重随机周期计量单位
  - 使用基于Z-Score的检测算法在线检测攻击
- 我们在Rocket-Chip上实现了末级缓存随机化
  - 2.84% 面积 (1.6% SRAM)，小于0.9% CPI, 2.1% 功耗

https://github.com/comparch-security/chipyard-random-llc

BSD-3 License，没有申请专利保护

欢迎来信： 宋威 songwei@iie.ac.cn

W. Song, Z. Xue, J. Han, Z. Li, P. Liu. “Randomizing set-associative caches against conflict-based cache side-channel attacks”. IEEE Transactions on Computers, vol. 73, no. 4, pp. 1019-1033, 2024

谢谢！

中国科学院信息工程研究所
INSTITUTE OF INFORMATION ENGINEERING,CAS