DAMARU: A Denial-of-Service Attack on Randomized Last-Level Caches

Pratik Kumar, Chavhan Sujeet Yashavant, and Biswabandan Panda

Abstract—Cross-core conflict-based timing attacks like Prime+Probe at the shared last-level cache (LLC) are practical and can cause information leakage. Cache address randomization is one of the techniques that claim to mitigate these attacks. CEASER, CEASER-S, and ScatterCache are the three recent randomized caches that use encryption engines to randomize the memory address mapping. CEASER and CEASER-S, along with encryption engines, remap the cache blocks periodically to break the static mapping of memory blocks into the LLC blocks. Encryption engine and remapping provide security to the randomized caches. However, access to encryption engines and the remapping of cache blocks are on the critical path of LLC accesses. We target encryption engine and remapping of randomized cache to mount a denial of service (DoS) attack named DAMARU. In DAMARU, the attacker frequently sends memory requests to the LLC that causes an increase in the victim's LLC access waiting time for the encryption engine. DAMARU is the first DoS attack on randomized caches where an attacker can cause a DoS even without thrashing the LLC. DAMARU provides a performance slowdown of up to 3.19X and 6X for 8-core and 16-core simulated systems, respectively. In terms of performance slowdown, the effectiveness of our DAMARU attack decreases with an increase in the number of encryption engines.

Index Terms— Cache memory, side-channel attacks, encryption

1 Introduction

On a multi-core system, the last-level cache (LLC) is shared among different cores. However, this sharing leads to different conflict-based attacks like Prime+Probe[1]. In conflict-based attacks, an attacker process creates an eviction set [1] by filling its data blocks into a cache set and by carefully evicting the victim's cache blocks. The reason behind such attacks is the static mapping of memory blocks into the LLC. Randomized caches such as CEASER[3], CEASER-S[4], and ScatterCache[5] break the traditional static mapping of the memory blocks to the cache sets. These techniques use an encryption engine that provides a random cache location based on a secret key. CEASER and CEASER-S provide a dynamic mapping of addresses to cache sets by periodically changing the secret key. ScatterCache is based on the skewed-associative cache[16] that randomly selects a skew to break the static mapping.

Randomized caches[3], [4], [5] claim to secure the LLC against conflict-based attacks with negligible performance overhead. While recent work [7] shows that the randomized caches are vulnerable to conflict-based attacks, we challenge the performance claims of these randomized caches. In this work, we use the non-linear ciphers, PRINCE as proposed by [7], and QARMA[5] that take eight and five cycles for encryption and decryption, respectively. CEASER and CEASER-S claim to have an average performance slowdown of 1% with Low-Latency Block Cipher (LLBC) [3], a linear cipher. ScatterCache claims to have an average performance slowdown of 2% with QARMA[6], a non-linear cipher.

The Problem. Encryption engine and the remapping process play a crucial role in the randomized caches, providing strong security guarantees. Both are on the critical path of LLC accesses and become the sources for a denial of service (DoS) attack.

Our Approach. We target the encryption engine implementation and remapping of randomized LLC to mount a DoS attack named DAMARU[1] on a multi-core. Our attacker frequently accesses the LLC and ensures that it gets ≈100% hit rate at the LLC. Through this process, our attacker denies the service of the encryption engine. This causes an increase in the victim's access waiting time for the encryption engine at the LLC. Frequent LLC accesses also invoke remapping and stall the LLC. The remapping process also evicts the victim's LLC blocks, increasing the victim's LLC miss count. Current DoS attack detectors[10] can not detect our attack as current detectors monitor the LLC misses, which is negligible for the DAMARU attacker. We choose four representative SPEC CPU 2017 benchmarks with low to high L2C and LLC misses per kilo instructions (MPKIs), as the victim processes. We provide a maximum slowdown of 3.19X and 6X with the state-of-the-art hardware prefetcher [14] on an 8-core and 16-core system, respectively. However, in terms of performance slowdown, the effectiveness of our DAMARU attack decreases with an increase in the #encryption engines.

Note that the primary goal of randomized caches is to provide security against conflict-based attacks, and these techniques are effective even in the presence of DAMARU attackers. In a nutshell, our contributions are as follows: (i) We propose a new class of DoS attack where a DoS attacker can be an LLC fitting application and the current DoS attack detectors can not detect. This is in contrast to previous DoS attacks [8], [10] where the attacker thrashes the LLC (Section 3). (ii) We evaluate the effectiveness of our attack on simulated multi-core systems. We quantify the major reasons for the performance slowdown (Section 4).

2 Background

Mitigations for conflict-based LLC attacks fall into two categories, namely randomization based [3], [4], [5], and cache partitioning based [17] and [18]. The partitioning based approach partitions the LLC among different cores and mitigates the cross-core conflict-based attacks. While the partitioning-based approaches can cause significant performance degradation [17], [18], randomization-based approaches claim to provide robust security guarantees with a marginal performance overhead, a win-win approach in terms of security and performance. Randomized LLCs can be used to provide security against cross-security domain conflict attacks, where a security domain can be a process, group of processes within a container, or a virtual machine. So, the proposed DAMARU attack is a threat to any system that uses randomized LLCs. The key idea of randomized caches is to map cache blocks to random sets dynamically. Encryption based randomization techniques are lightweight in terms of hardware storage. Some of the techniques that implement encryption engine based randomization are CEASER, CEASER-S, and ScatterCache.

CEASER encrypts a physical address based on a key to get the encrypted address on an LLC access. CEASER uses two encryption engines per LLC slice. It chooses a cache set based on this encrypted address. It also remaps a cache block after every 100 LLC accesses to ensure that the attacker cannot create an eviction

- *Pratik Kumar and Biswabandan Panda are with the IIT Bombay, Mumbai, Maharashtra 400076, India. E-mail: {pratikkr, biswa}@cse.iitb.ac.in.*
- *Chavhan Sujeet Yashavant is with IIT Kanpur, Kanpur, Uttar Pradesh 208016, India. E-mail: sujeetc@cse.iitk.ac.in.*

Manuscript received 12 July 2021; revised 19 Aug. 2021; accepted 9 Sept. 2021. Date of publication 14 Sept. 2021; date of current version 4 Oct. 2021.
This work was supported by the SRC Under Grant SRC-2853.001.
(Corresponding author: Biswabandan Panda.)
Digital Object Identifier no. 10.1109/LCA.2021.3112180

1. DAMARU is a two-headed drum. In this paper, two heads represent the confidentiality and the availability aspects of security.

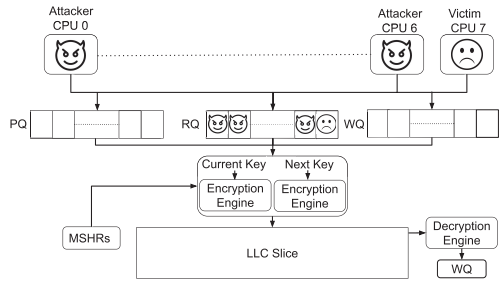

Fig. 1. DoS at the encryption engine with CEASER. PQ/RQ/WQ:prefetch/read/write queues. MSHRs stands for miss status holding registers.

set. CEASER uses LLBC cipher that takes two cycles for encryption and decryption. It claims to be secure against [1], which used to be the best state-of-the-art algorithm for eviction set creation. However, Vila *et al*. proposed a faster eviction set creation [2] which breaks CEASER. To tolerate such attacks, CEASER needs to remap frequently, which brings a lot of performance overhead.

CEASER-S and ScatterCache are based on the skewed cache[16] that does way-based partitioning, where each partition has a different mapping function. CEASER-S and ScatterCache use four and one encryption engines, respectively. These designs randomly select a partition to fill a cache block, and the partition's hash function provides a set number. As LLBC is not secure, and instead, a non-linear PRINCE [9] can be used with CEASER and CEASER-S.

Listing 1. DAMARU Attacker Code Snippet

```
1  //NUM_SLICES : Number of LLC slices
2  //MAX_PRIVATE_ASSOC : Maximum L1/L2C associativity
3  //NUM_L2C_SET : Number of cache sets in L2C
4  //arr : array of size equals to LLC
5  //array_element_size : 8 bytes
6  chunk_size = cache_block_size/array_element_size
7  bits_in_control= Page_Offset-Block_Offset
8  blocks_in_control = 2^bits_in_control
9  blocks_not_in_control= 2^L2C_index_bits-bits_in_control
10 while(1) {
11    for(i=1 to NUM_SLICES)
12     for(j=1 to MAX_PRIVATE_ASSOC+1)
13       for(k=1 to blocks_not_in_control)
14          x = arr[((j×L2C_SET)+(k×blocks_in_control)+
                i)×chunk_size]
15 }
```

3 DAMARU Attack

Attacker's Goal. A encryption engine(s), which are placed beside an LLC slice, can be non-pipelined or pipelined. On a non-pipelined encryption engine, a major (one cycle less than the encryption latency) portion of the encryption latency becomes unused due to non-overlapped execution. We exploit this observation and send frequent LLC accesses that hogs the encryption engine. As a result, it also hogs the micro-architectural queues (for example, the read queue). This process increases the LLC access time of the victim as it waits for the encryption engine, as shown in Fig. 1. Also, frequent accesses at the LLC frequents the remap activity, which stalls the LLC further and causes an increase in the waiting time.

Attacker's Approach. Encryption engine and remapping are the reasons for the security of a randomized cache, and both are on the critical path of LLC access. Our attacker exploits both by frequently accessing the LLC and ensuring a ≈100% LLC hit rate and ≈100% miss rate at the L1D and L2C. It is easy to mount our DoS attack as the attacker's private caches are in control of the attacker. So, it is easy to get a ≈100% miss rate at the private caches. The reason behind the high LLC hit rate is that a DAMARU attacker accesses a small number of blocks (100 blocks per slice) repeatedly at the LLC, leading to a shorter reuse distance for each of these blocks. Hence, the attacker's blocks still reside in the LLC even in the presence of remapping and randomization. We have also performed DAMARU with the state-of-the-art [11], [12], [13] replacement policies at LLC, and observe a similar trend.

DAMARU Attacker's Code Snippet. Listing 1 shows our attacker code for a randomized LLC of 16MB (8 2MB LLC slices). The replacement policy at the private caches is LRU. Our attacker code is generic and independent of the size of the OS page. Our attacker code provides a ≈100% miss rate at L1D and L2C and a ≈100% hit rate at LLC. We *unroll* all the *for* loops(line nos. 11,12, and 13) to get a higher L1D and L2C MPKI of 998. Our attacker code iterates through *MAX_PRIVATE_ASSOC* + 1 (line no. 12) at a block size distance of L2C_SET to map a block into the same L2C set and to make every access to this set a miss. The attacker also iterates through each LLC slice (line no. 11) to increase the victim's LLC access waiting time for all the slices. As the page size bits do not overlap with all the L2C index bits hence all the *blocks_not_in_control* (line no. 13) are iterated through. For a 4KB OS page with an L2C size of 512KB (8 way), four bits are not in control of the attacker. So the attacker has to iterate sixteen times (line no. 13), and for a 2MB page, blocks not in control become zero as every bit is in the control of the attacker.

TABLE 1
Parameters of the Simulated System

Processor	4 GHz, out of order
L1D, L1I, L2C	48KB (12 way, IPCP [14]), 32KB (8 way), 512KB (8 way, IPCP [14])
Sliced LLC	16MB (2MB per slice), 16 way
MSHRs	16, 16, 32, 32/256/512 MSHRs
Replacement Policy	at L1D, L1I, L2C, LLC with 1/8/16 cores LRU at L1 and L2, SRRIP[13] at the LLC, random policy for ScatterCache
DRAM Controller	1600 MHz (11-11-11), 12.8 GB/sec

4 Evaluation

We show the effectiveness of our attack on all three randomized caches, i.e., CEASER, CEASER-S, and ScatterCache. We use a trace-based Champsim[15] simulator. We implement 12-round PRINCE and QARMA as encryption engines, with eight [7] and five cycle[5] latency, respectively. Table 1 shows the parameters of the simulated system. We choose four representative SPEC CPU 2017 benchmarks with low to high L2C and LLC MPKIs (Table 2) for a 512KB L2C and a 2MB LLC. We simulate their respective region of interests. We use an OS page size of 2MB, similar to [1]. We simulate an eight-core system with all possible combinations of attackers and victims. We observe that the slowdown increases as we increase the number of attackers. We report performance slowdown normalized to baseline non-randomized cache with the same number of victims and attackers running on the multi-core system.

Performance Slowdown With Randomized LLC. We observe a maximum slowdown of 3.19X with IPCP prefetcher and 2X without IPCP with seven attackers and one victim running on an 8-core

TABLE 2
L2C/LLC MPKIs of Representative Benchmarks

Benchmark	L2C MPKI	LLC MPKI
perl	0.66	0.66
xalan	3.2	1.8
gcc	70.05	70.04
mcf	153.1	146.9

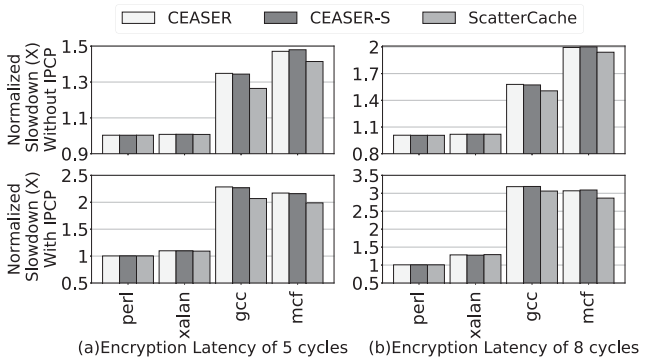

Fig. 2. Normalized slowdown due to randomization of victim core with CEASER, CEASER-S, and ScatterCache.

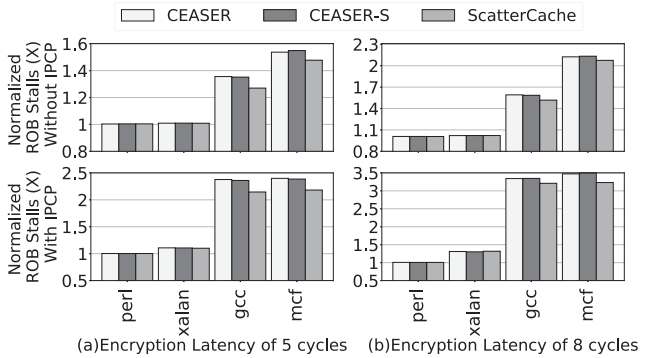

Fig. 3. Normalized ROB stalls for victim core with CEASER, CEASER-S and ScatterCache.

CPU. Fig. 2 shows performance slowdown normalized to non-randomized LLC on an 8-core system with seven attackers. High L2C MPKI benchmarks show a higher slowdown. In comparison, benchmarks with low L2C MPKI show a lower slowdown because the attacker's impact varies with the number of victims' LLC accesses. All three randomized caches show different slowdowns due to differences in their cache mapping techniques. On a 16-core simulated system with 15 attackers, we observe a maximum slowdown of 6X with IPCP.

Performance Slowdown With Non-Randomized LLC. We perform our DoS attack with seven attackers on a non-randomized LLC without an encryption engine, and the maximum, minimum, and average performance slowdowns that we get are 1.13X,1.0X, and 1.06X, respectively. This is intuitive as the contention at the LLC due to the attacker is negligible with the non-randomized cache. *This shows that our DoS attack is specific to randomized LLCs.*

Effect on ROB Stalls. Fig. 3 shows the increase in normalized reorder buffer (ROB) stalls due to DAMARU for randomized LLCs. With DAMARU, the victim's waiting time for LLC accesses increases. The additional waiting time for L2C misses, leading to stalling at the head of the ROB. There is a strong correlation between Figs. 2 and 3, which is intuitive.

The Hardware Prefetcher Effect. In the randomized cache, slowdown increases with the hardware prefetcher at the L1D and L2C compared to no prefetching. As RQ, WQ, PQ, and MSHR share the encryption engine, and demand requests have higher priority over prefetch requests; with randomization, prefetcher timeliness gets affected that affects the overall performance. This delay in processing prefetch requests further denies demand requests from occupying entries in MSHRs at L1D and L2C. Overall, the DAMARU attacker is more effective in the presence of a hardware prefetcher; as prefetch requests get late, MSHR occupancy with prefetch requests becomes longer, which affects the demand miss latencies at the L1D and L2C.

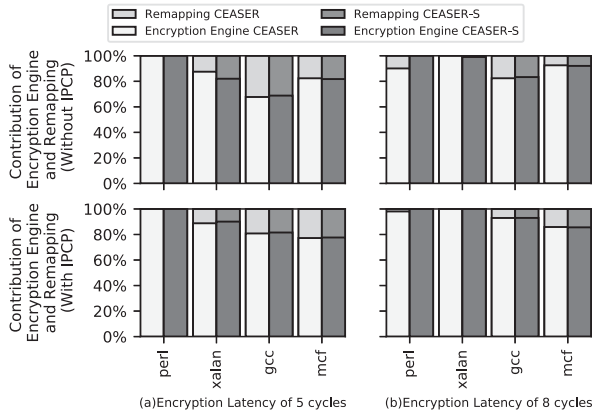

Fig. 4. Contribution of encryption engine and remapping in the performance slowdown with CEASER and CEASER-S with different encryption latencies.

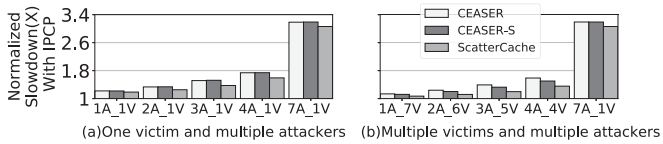

Fig. 5. Performance slowdown with different combinations of attackers(A) and victims(V) on an 8-core system.

To understand the effect of prefetcher in more detail, we study the impact of IPCP prefetcher at L1D and L2C for `gcc` benchmark. With DAMARU, L1D prefetch coverage, accuracy, and timeliness drop by 19%, 15%, and 22% compared to the baseline system, while L2C prefetch coverage, accuracy, and timeliness drop significantly by 49%, 88%, and 81%, respectively. Note that `gcc` is a prefetch friendly benchmark that provides significant performance improvement (more than 3.5X) with IPCP [14]. Though `mcf` shows the highest slowdown among all the chosen benchmarks without prefetcher, with prefetcher On, the performance of `gcc` is worst affected as it is prefetcher friendly benchmark.

Contribution of Encryption Engine and Remapping. We analyze the contribution of the encryption engine and remapping to the performance slowdown for CEASER and CEASER-S. Note that ScatterCache does not perform remapping. Fig. 4 shows that the encryption engine contributes significantly to the performance slowdown. We quantify the contribution of the remapping process and encryption engines.

Effect of Multiple DAMARU Attackers. Fig. 5 shows the maximum slowdown with the victim (`gcc`) for different combinations of attackers and victims running on an 8-core system. We choose `gcc` as it shows the maximum performance slowdown with the IPCP prefetcher (Fig. 2). We can see that an increase in the number of attackers increases the slowdown due to increased LLC accesses.

Effect of Number of Encryption Engines. For demand/prefetch reads, writes, and coherence messages, we use two encryption engines (one for the current key and one for the next key)[3] for CEASER; similarly, four encryption engines (two per partition) [4] for CEASER-S, and one encryption engine [5] for ScatterCache. We further study DAMARU with more number of encryption engines. Fig. 6 shows that the victim's (`gcc`) slowdown in the presence of seven attackers decreases with an increase in the number of encryption engines. Across all the randomized LLCs, eight to 16 encryption engines per LLC slice can decrease DAMARU's performance slowdown.

Effect of Pipelined Encryption Engine. A non-pipelined encryption engine is one of the primary reasons for the performance slowdown; hence we evaluate DAMARU for a pipelined encryption

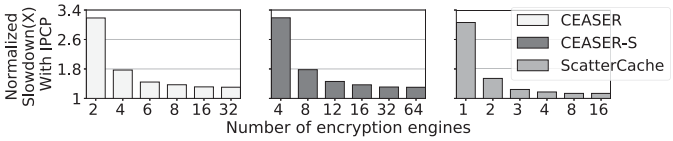

Fig. 6. Performance slowdown with #encryption engines.

engine with an encryption latency of three cycles and five cycles (an improvement from five and eight cycles). CEASER, CEASER-S, and ScatterCache show a maximum slowdown of 2.34X, 2X, and 1.84X with prefetcher, and 1.69X, 1.41X, and 1.14X without prefetcher, respectively. The remapping process is the primary contributor to the slowdown as it evicts the victim's blocks and significantly increases the victim's LLC miss count. We observe that the impact on conflict misses due to remapping is significant with four attackers and four victims as with an increase in the number of victims the conflict misses also increases. Due to this, we have observed maximum slowdown with four attackers and four victims for the pipelined encryption engine. In the case of ScatterCache, a random replacement policy at the LLC increases the performance slowdown, which further worsens in the presence of a prefetcher. With the encryption engine latency of five cycles, there is a ≈5% increase in the performance slowdown compared to the encryption engine latency of three cycles.

Mitigation Techniques. To mitigate DAMARU, a detector based approach can be used as the DAMARU attacker has a ≈100% miss rate at the L1D and L2C, and ≈100% hit rate at the LLC (unusual for benign applications). A detector can be designed based on these metrics as per performance counters, and the OS can de-schedule or migrate to a different socket. The other approach would be to have many encryption engines (eight to 16 per LLC slice, a costly solution). Also, reducing the rate of remapping without affecting the security guarantees can reduce the performance slowdown.

5 Conclusion

We proposed a new denial of service attack named DAMARU on randomized LLC where the attacker is an LLC fitting application and that current DoS attack detectors can not detect our proposed attack. We target the encryption engine and its latency, remapping, and random replacement policy to cause a performance slowdown of 3.19X and 6X on 8-core and 16-core multicore systems, respectively. We discuss the primary reasons for the significant performance slowdown. We also perform sensitivity studies in terms of encryption engines and number of attackers and victims. We find that the effectiveness of our attack decreases with an increase in the number of encryption engines.

Acknowledgments

The authors would like to thank members of CARS group and Vinod Ganesan for their valuable feedback on the initial draft.

References

[1] F. Liu, Y. Yarom, Q. Ge, G. Heiser, and R. B. Lee, "Last-level cache side-channel attacks are practical," in *Proc. IEEE Symp. Secur. Privacy*, 2015, pp. 605–622.
[2] P. Vila, B. Köpf, and J. F. Morales, "Theory and practice of finding eviction sets," in *Proc. IEEE Symp. Secur. Privacy*, 2019, pp. 39–54.
[3] M. K. Qureshi, "CEASER: Mitigating conflict-based cache attacks via encrypted-address and remapping," in *Proc. 51st Annu. IEEE/ACM Int. Symp. Microarchit.*, 2018, pp. 775–787.
[4] M. K. Qureshi, "New attacks and defense for encrypted-address cache," in *Proc. ACM/IEEE 46th Annu. Int. Symp. Comput. Archit.*, 2019, pp. 360–371.
[5] M. Werner *et al.*, "ScatterCache: Thwarting cache attacks via cache set randomization," in *Proc. 28th USENIX Secur. Symp.*, 2019, pp. 675–692.
[6] R. Avanzi, "The QARMA block cipher family," *IACR Trans. Symmetric Cryptol.*, vol. 2017, no. 1, pp. 4–44, 2017.
[7] R. Bodduna, V. Ganesan, P. Slpsk, K. Veezhinathan, and C. Rebeiro, "Brutus: Refuting the security claims of the cache timing randomization countermeasure proposed in CEASER," *IEEE Comput. Archit. Lett.*, vol. 19, no. 1, pp. 9–12, Jan.–Jun. 2020.
[8] T. Moscibroda and O. Mutlu, "Memory performance attacks: Denial of memory service in multi-core systems," in *Proc. 16th USENIX Secur. Symp.*, 2007, pp. 1–18.
[9] J. Borghoff *et al.*, "PRINCE – A low-latency block cipher for pervasive computing applications," in *Proc. Int. Conf. Theory Appl. Cryptol. Inf. Secur.*, 2012, pp. 208–225.
[10] M. Bechtel and H. Yun, "Denial-of-service attacks on shared cache in multicore: Analysis and prevention," in *Proc. IEEE Real-Time Embedded Technol. Appl. Symp.*, 2019, pp. 357–367.
[11] A. Jain and C. Lin, "Back to the future: Leveraging Belady's algorithm for improved cache replacement," *ACM SIGARCH Comput. Archit. News*, vol. 44, no. 3, pp. 78–89, 2016.
[12] C. -J. Wu, A. Jaleel, W. Hasenplaugh, M. Martonosi, S. C. Steely, and J. Emer, "SHiP: Signature-based hit predictor for high performance caching," in *Proc. 44th Annu. IEEE/ACM Int. Symp. Microarchit.*, 2011, pp. 430–441.
[13] A. Jaleel *et al.*, "High performance cache replacement using re-reference interval prediction (RRIP)," in *Proc. 37th Annu. Int. Symp. Comput. Archit.*, 2010, pp. 60–71.
[14] S. Pakalapati and B. Panda, "Bouquet of instruction pointers: Instruction pointer classifier-based spatial hardware prefetching," in *Proc. ACM/IEEE 47th Annu. Int. Symp. Comput. Archit.*, 2020, pp. 118–131.
[15] Champsim simulator, 2015. [Online]. Available: https://github.com/ChampSim/ChampSim
[16] A. Seznec, "A case for two-way skewed-associative caches," in *Proc. 20th Annu. Int. Symp. Comput. Archit.*, 1993, pp. 169–178.
[17] C. A. T. Intel, "Improving real-time performance by utilizing cache allocation technology," Intel Corporation, pp. 1–16, Apr. 2015.
[18] V. Kiriansky, I. Lebedev, S. Amarasinghe, S. Devadas, and J. Emer, "DAWG: A defense against cache timing attacks in speculative execution processors," in *Proc. 51st Annu. IEEE/ACM Int. Symp. Microarchit.*, 2018, pp. 974–987.
[19] SPEC CPU 2017 traces, 2017. [Online]. Available: http://hpca23.cse.tamu.edu/champsim-traces/speccpu/