On Differential Privacy and Caching

Badih Ghazi[*] Ravi Kumar[†] Pasin Manurangsi[‡]
Jelani Nelson[§] Adam Sealfon[▽] Samson Zhou[¶]

Abstract

We study caching algorithms from the perspective of protecting sensitive information against side-channel attacks. Traditional caching algorithms such as Least Recently Used (LRU) and Marker reveal access patterns, potentially exposing confidential information. Motivated by these risks, we investigate differentially private (DP) caching mechanisms, introducing and analyzing notions of sequence-DP (that aims to protect the sequence of cache misses) and cost-DP (that aims to protect the number of cache misses) for caching algorithms. We show that achieving sequence-DP can lead to prohibitively high cache miss rates, which motivates the weaker notion of cost-DP. We develop randomized caching algorithms, including variants of LRU and Marker, that achieve polylogarithmic competitive ratios under cost-DP guarantees. Additionally, we prove that pure-cost-DP is unattainable for caching algorithms, reinforcing the need for approximate-cost-DP. Our results highlight the trade-offs between privacy and efficiency in online caching, providing both lower bounds and new algorithms that balance these objectives.

1 Introduction

Online caching is a fundamental algorithmic problem in computer architecture, database systems, distributed systems, operating systems, and theoretical computer science. In the online caching problem, a system must manage a limited-size cache while serving a sequence of page requests that arrive over time, making eviction decisions without knowledge of future requests. The objective is typically to minimize the number of cache misses, and the problem is often analyzed through the lens of competitive analysis, comparing the performance of an online algorithm against an optimal offline strategy with full knowledge of the sequence of data requests. Classical caching algorithms are designed to minimize cache misses by exploiting temporal locality of page requests.

Cache hits and misses take different amounts of time due to the varying latencies associated with accessing data from the cache versus slower memory or storage. These timing differences can be exploited by a malicious observer, who can measure the varying latencies in data access patterns to infer sensitive information. For example, a timing attack can use these latencies to infer which memory locations are being accessed to target cryptographic operations, where the time taken can depend on the specific key or input, so that even small timing differences can allow attackers to deduce cryptographic keys or plaintexts. Timing attacks that can reveal cache behavior and the frequency of cache misses may also expose properties of the computation such as the size of the working set or the application's locality of reference, which may be sensitive. Timing attacks are particularly effective in shared environments, such as cloud computing, where attackers can passively observe system resource usages or memory access patterns without direct access to the target's data [YF14, GSM15, LYG+15, CCX+20, LSG+20, DNA+21, PGGV21, AOK+22, CDBY22, RNLY22, TKK+22]. These

[*]Google Research. E-mail: badihghazi@gmail.com.
[†]Google Research. E-mail: ravi.k53@gmail.com.
[‡]Google Research. E-mail: pasin@google.com.
[§]UC Berkeley and Google Research. E-mail: minilek@alum.mit.edu.
[▽]Google Research. E-mail: adamsealfon@google.com.
[¶]Texas A&M University. E-mail: samsonzhou@gmail.com.

types of attacks highlight a broader challenge: even indirect observations of system behavior, such as timing or access patterns, can leak sensitive information about users or workloads.

Motivated by this risk of indirect information leakage through cache access patterns, we study mechanisms to mitigate the exposure of sensitive information in cache systems, with the goal to ensure that individual access patterns remain indistinguishable even in the presence of timing-based side-channel observations. In particular, we use differential privacy (DP) [DMNS06], which has emerged as the gold standard for mathematically guaranteeing the outcome of a computation is nearly the same whether or not any single individual's data is included, to limit the amount of information an attacker can infer about a specific data request from the entire sequence of cache access patterns.

1.1 Our Contributions

Let $[n] = \{1, \ldots, n\}$ be the universe of pages and let $\mathcal{R} = r_1, \ldots, r_{|\mathcal{R}|}$ be a page request sequence, where $r_t \in [n]$ for $t \in [|\mathcal{R}|]$. We will generally set $T = |\mathcal{R}|$ when there is no ambiguity. Let k be the cache size; a caching algorithm holds at most k pages in its cache at any time. A caching algorithm $\mathcal{A}$ processes the page request r_t. If r_t is not in its cache (aka *cache miss*), then it loads r_t into its cache, possibly evicting some other page to obey the cache size constraint.[1] For a caching algorithm $\mathcal{A}$ and a page request sequence $\mathcal{R}$, let $\mathsf{seq}_{\mathcal{A}}(\mathcal{R})$ denote the corresponding binary sequence of length $|\mathcal{R}|$ in which the t-th bit is 1 if and only if a cache miss occurred while processing r_t.

Sequence- and Cost-DP. Two page request sequences $\mathcal{R} = r_1, \ldots, r_{|\mathcal{R}|}$ and $\mathcal{R}' = r'_1, \ldots, r'_{|\mathcal{R}'|}$ are *neighboring* if $|\mathcal{R}| = |\mathcal{R}'|$ and they differ in the identity of a single page request, i.e., $r_t = r'_t$ for all but at most one $t \in [|\mathcal{R}|]$; we denote this $\mathcal{R} \sim \mathcal{R}'$. A natural requirement for a private caching algorithm is that $\mathsf{seq}_{\mathcal{A}}(\mathcal{R})$ and $\mathsf{seq}_{\mathcal{A}}(\mathcal{R}')$ are somewhat indistinguishable for $\mathcal{R} \sim \mathcal{R}'$. We formalize this next.

Definition 1.1 (Sequence-DP)**.** *A caching algorithm $\mathcal{A}$ is* (ε, δ)-sequence differentially private ((ε, δ)-seqDP) *if for every pair $\mathcal{R} \sim \mathcal{R}'$ of neighboring page request sequences, it holds that*

$$\mathbf{Pr}\left[\mathsf{seq}_{\mathcal{A}}(\mathcal{R}) \in S\right] \leq e^{\varepsilon} \cdot \mathbf{Pr}\left[\mathsf{seq}_{\mathcal{A}}(\mathcal{R}') \in S\right] + \delta, \qquad \forall S \subseteq \{0,1\}^{|\mathcal{R}|}.$$

Unfortunately, we show it is not possible to achieve sequence-DP while still providing meaningful guarantees about the number of cache misses, as formalized below.

Theorem 1.2. *Assume that $n \geq T$. For any $\varepsilon, \delta \geq 0$ and a sequence of T page requests, any caching algorithm that is (ε, δ)-seqDP must have $(e^{-\varepsilon} - \delta) \cdot T$ cache misses in expectation.*

For $\varepsilon = O(1), \delta = o(1)$, Theorem 1.2 states that any (ε, δ)-seqDP caching algorithm must have a linear number of cache misses in expectation, regardless of the sequence of page requests. Note that it is trivial to achieve T cache misses, and we cannot improve on this trivial bound for *any* sequence of page requests. In particular, for a cache of size k (where $k \ll T$), there exists a sequence of page requests that would cause k cache misses under an optimal policy. However, a caching algorithm might be forced to incur $\Omega(T)$ cache misses for this same sequence, purely to satisfy the sequence-DP property. In other words, the definition of sequence-DP seems incompatible with the goal of designing caching algorithms with good competitive ratio.

To circumvent this, we define the following notion of cost-DP for caching algorithms, where we use $\mathsf{cost}_{\mathcal{A}}(\mathcal{R})$ to denote the number of cache misses by an algorithm $\mathcal{A}$ on page request sequence $\mathcal{R}$, i.e., $\mathsf{cost}_{\mathcal{A}}(\mathcal{R}) = \|\mathsf{seq}_{\mathcal{A}}(\mathcal{R})\|_1$.

Definition 1.3 (cost-DP)**.** *A caching algorithm $\mathcal{A}$ is* (ε, δ)-cost differentially private ((ε, δ)-costDP) *if for every pair $\mathcal{R} \sim \mathcal{R}'$ of neighboring page request sequences, it holds that*

$$\mathbf{Pr}\left[\mathsf{cost}_{\mathcal{A}}(\mathcal{R}) \in C\right] \leq e^{\varepsilon} \cdot \mathbf{Pr}\left[\mathsf{cost}_{\mathcal{A}}(\mathcal{R}') \in C\right] + \delta, \qquad \forall C \subseteq \mathbb{Z}.$$

[1] We allow our algorithms to incur "dummy" cache misses, i.e., they are allowed to evict any page in the cache and reload it immediately, just to incur a cache miss.

Main Results. Our main technical contribution for cost-DP is a simple transformation that can make any caching algorithm cost-DP, with a small overhead in the cache miss rate. To state this, we say that a caching algorithm is (α, β)-*competitive* if its cost is at most α times the offline optimum plus β, i.e., α is the (multiplicative) competitive ratio and β is the additive factor; if $\beta = 0$, we say it is α-*competitive*. (See **??** for a precise definition.) With this in mind, our transformation can be stated as follows.

Theorem 1.4. *Given an algorithm that is α-competitive with probability $1 - \frac{\delta}{T}$, there exists an (ε, δ)-costDP algorithm that is $\left(O(\alpha), O\left(\frac{\alpha k}{\varepsilon} + \frac{\log \frac{T}{\delta}}{\varepsilon^2}\right) \cdot \log \frac{T}{\delta}\right)$-competitive on any sequence of T page requests, with probability at least $1 - \delta$.*

In other words, we can make any caching policy satisfy (ε, δ)-costDP while retaining the same competitive ratio and incurring an additive overhead that depends only linearly on k and logarithmically on $T, 1/\delta$.

The well-known Marker algorithm is $O\left(\log \frac{kT}{\delta}\right)$-competitive with probability at least $1 - \frac{\delta}{T}$. (See **??** for more details.) Combining with Theorem 1.4 we get:

Corollary 1.5. *There exists an (ε, δ)-costDP algorithm that is $\left(O\left(\log \frac{kT}{\delta}\right), O\left(\frac{k \log^2 \frac{kT}{\delta}}{\varepsilon} + \frac{\log^2 \frac{T}{\delta}}{\varepsilon^2}\right)\right)$-competitive on any sequence of T page requests, with probability at least $1 - \delta$.*

On the other hand, Marker is less commonly used in practice compared to LRU and its variants. We analyze the competitive ratio of a probabilistic version of LRU where each item is increasingly likely to be evicted from the cache as its rank among the recently requested pages increases.

Theorem 1.6. *Randomized LRU is $O\left(\log^2 k\right)$-competitive on any sequence of T page requests with probability 0.99 and $O\left(\log^2 k \log \frac{T}{\delta}\right)$ competitive with probability $1 - \frac{\delta}{\mathrm{poly}(T)}$.*

Together, Theorem 1.4 and Theorem 1.6 imply that randomized LRU can be used to achieve an (ε, δ)-costDP algorithm that is $\left(O\left(\log^3 \frac{kT}{\delta}\right), O\left(\frac{k \log^3 \frac{kT}{\delta}}{\varepsilon} + \frac{\log^2 \frac{T}{\delta}}{\varepsilon^2}\right)\right)$-competitive, although we note that this is asymptotically worse than that of the (transformed) Marker algorithm.

Finally, we remark that Theorem 1.4 and its consequences provide guarantees for approximate-DP. A natural question is to ask whether the statements could be strengthened by achieving pure-cost-DP with non-trivial guarantees on the number of cache misses. We show this is not possible:

Theorem 1.7. *For any $\varepsilon \geq 0$ and any sequence of T page requests, any caching algorithm that is ε-costDP must have T cache misses with probability one.*

Motivation. We describe two scenarios that highlight the differences between cost-DP and sequence-DP. These scenarios clarify when the weaker but achievable definition of cost-DP may be appropriate.

Scenario 1: Cloud computing. Consider a trusted server that performs delegated computations on behalf of multiple clients. Individual clients may be adversarial and contend for shared system resources, allowing them to observe the server's CPU usage and network traffic. However, since the server is trusted, the adversary cannot directly observe memory accesses or values stored in memory during the computation.

In this setting, the main leakage is the total execution time, as measured either by monitoring CPU usage or from the timestamps of requests on the network. For many computations, runtime is dominated by the number of cache misses due to the high latency of main memory accesses [WM95, GYK+24]. If executions with the same number of cache misses have similar runtimes, then enforcing cost-DP effectively limits timing leakage. More generally, we can combine the timing privacy framework [RV24] with cost-DP to provide provable protection against timing-based side channels while still maintaining the performance benefits of workloads with favorable cache behavior.

Scenario 2: Shared memory. Now consider computations performed on a system with shared memory that may be visible to other processes running on the same system. For example, such a configuration is typical of Trusted Execution Environments (TEEs; see [MRRL23] for a recent survey), where the TEE may contain a small secure cache but accesses to shared main memory may be observable to an adversary running other processes on the same machine. In this scenario, the leakage consists of the cache misses and their timing, as well as the values stored in main memory.

We can reduce this leakage using Oblivious RAM (ORAM) [Gol87, Ost90, GO96], a cryptographic technique that hides memory access patterns by randomizing data storage and retrieval [SvDS+18, AKL+23]. While ORAM conceals the order and identity of memory accesses, it incurs a significant performance overhead and sacrifices the performance benefits from caching repetitive or predictable memory access patterns. A natural workaround is to apply ORAM only to main memory accesses and not for cache accesses, since the cache is not in shared memory and cache hits remain unobserved by the adversary.

This approach, along with encrypting the values stored in memory, hides both the accessed memory locations and their values. However, it does not yet conceal the number or pattern of cache misses. We can combine the approach with sequence-DP to ensure that the sequence of cache misses follows a similar distribution for any pair of neighboring request sequences.

In contrast, cost-DP provides incomplete protection in this scenario. While it masks the total count of cache hits and misses, it does not hide their interleaving or their timing, which could reveal information about the input. For example, an execution with cache misses concentrated at the start and one with misses at the end could be distinguished by an adversary that observes the timing of the cache misses, even with cost-DP. (While this specific attack may not apply to our proposed algorithm, the definition of cost-DP does not rule it out.) Nevertheless, while cost-DP does not provide a robust standalone guarantee here, it reduces the attack surface and may remain useful in shared memory settings when combined with additional mitigations such as batching and randomized delays.

References

[AKL+23] Gilad Asharov, Ilan Komargodski, Wei-Kai Lin, Kartik Nayak, Enoch Peserico, and Elaine Shi. Optorama: Optimal oblivious RAM. *J. ACM*, 70(1):4:1–4:70, 2023. 4

[AOK+22] Ayush Agarwal, Sioli O'Connell, Jason Kim, Shaked Yehezkel, Daniel Genkin, Eyal Ronen, and Yuval Yarom. Spook.js: Attacking chrome strict site isolation via speculative execution. In *43rd IEEE Symposium on Security and Privacy, SP*, pages 699–715, 2022. 1

[CCX+20] Guoxing Chen, Sanchuan Chen, Yuan Xiao, Yinqian Zhang, Zhiqiang Lin, and Ten-Hwang Lai. Sgxpectre: Stealing intel secrets from SGX enclaves via speculative execution. *IEEE Secur. Priv.*, 18(3):28–37, 2020. 1

[CDBY22] Jack Cook, Jules Drean, Jonathan Behrens, and Mengjia Yan. There's always a bigger fish: a clarifying analysis of a machine-learning-assisted side-channel attack. In *ISCA '22: The 49th Annual International Symposium on Computer Architecture*, pages 204–217, 2022. 1

[DMNS06] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam D. Smith. Calibrating noise to sensitivity in private data analysis. In *Theory of Cryptography, Third Theory of Cryptography Conference, TCC, Proceedings*, pages 265–284, 2006. 2

[DNA+21] Sankha Baran Dutta, Hoda Naghibijouybari, Nael B. Abu-Ghazaleh, Andres Marquez, and Kevin J. Barker. Leaky buddies: Cross-component covert channels on integrated CPU-GPU systems. In *48th ACM/IEEE Annual International Symposium on Computer Architecture, ISCA*, pages 972–984, 2021. 1

[GO96] Oded Goldreich and Rafail Ostrovsky. Software protection and simulation on oblivious rams. *J. ACM*, 43(3):431–473, 1996. 4

[Gol87] Oded Goldreich. Towards a theory of software protection and simulation by oblivious rams. In *Proceedings of the 19th Annual ACM Symposium on Theory of Computing*, pages 182–194, 1987. 4

[GSM15] Daniel Gruss, Raphael Spreitzer, and Stefan Mangard. Cache template attacks: Automating attacks on inclusive last-level caches. In *24th USENIX Security Symposium, USENIX Security*, pages 897–912, 2015. 1

[GYK+24] Amir Gholami, Zhewei Yao, Sehoon Kim, Coleman Hooper, Michael W. Mahoney, and Kurt Keutzer. Ai and memory wall. *IEEE Micro*, 44(3):33–39, 2024. 3

[LSG+20] Moritz Lipp, Michael Schwarz, Daniel Gruss, Thomas Prescher, Werner Haas, Jann Horn, Stefan Mangard, Paul Kocher, Daniel Genkin, Yuval Yarom, Mike Hamburg, and Raoul Strackx. Meltdown: reading kernel memory from user space. *Commun. ACM*, 63(6):46–56, 2020. 1

[LYG+15] Fangfei Liu, Yuval Yarom, Qian Ge, Gernot Heiser, and Ruby B. Lee. Last-level cache side-channel attacks are practical. In *2015 IEEE Symposium on Security and Privacy, SP*, pages 605–622, 2015. 1

[MRRL23] Antonio Muñoz, Ruben Ríos, Rodrigo Román, and Javier López. A survey on the (in)security of trusted execution environments. *Computers & Security*, 129:103180, 2023. 4

[Ost90] Rafail Ostrovsky. Efficient computation on oblivious rams. In *Proceedings of the 22nd Annual ACM Symposium on Theory of Computing*, pages 514–523, 1990. 4

[PGGV21] Antoon Purnal, Lukas Giner, Daniel Gruss, and Ingrid Verbauwhede. Systematic analysis of randomization-based protected cache architectures. In *42nd IEEE Symposium on Security and Privacy, SP*, pages 987–1002, 2021. 1

[RNLY22] Joseph Ravichandran, Weon Taek Na, Jay Lang, and Mengjia Yan. PACMAN: attacking ARM pointer authentication with speculative execution. In *ISCA '22: The 49th Annual International Symposium on Computer Architecture*, pages 685–698, 2022. 1

[RV24] Zachary Ratliff and Salil Vadhan. A framework for differential privacy against timing attacks, 2024. 3

[SvDS+18] Emil Stefanov, Marten van Dijk, Elaine Shi, T.-H. Hubert Chan, Christopher W. Fletcher, Ling Ren, Xiangyao Yu, and Srinivas Devadas. Path ORAM: an extremely simple oblivious RAM protocol. *J. ACM*, 65(4):18:1–18:26, 2018. 4

[TKK+22] Youssef Tobah, Andrew Kwong, Ingab Kang, Daniel Genkin, and Kang G. Shin. Spechammer: Combining spectre and rowhammer for new speculative attacks. In *43rd IEEE Symposium on Security and Privacy, SP*, pages 681–698, 2022. 1

[WM95] Wm A Wulf and Sally A McKee. Hitting the memory wall: Implications of the obvious. *ACM SIGARCH computer architecture news*, 23(1):20–24, 1995. 3

[YF14] Yuval Yarom and Katrina Falkner. FLUSH+RELOAD: A high resolution, low noise, L3 cache side-channel attack. In *Proceedings of the 23rd USENIX Security Symposium*, pages 719–732, 2014. 1