Access Cost Tradeoffs in Mixed-Mode Oblivious RAM

Wojciech Ciszewski
University of Warsaw
IDEAS Institute
ORCID: 0000-0002-5652-6066
w.ciszewski2@uw.edu.pl

Emanuele Ragnoli
University of Warsaw
ORCID: 0000-0002-6310-091X
emanuele.ragnoli@gmail.com

Abstract

A mixed-mode oblivious RAM (ORAM) lets each memory request be marked *public* or *private*: public requests disclose their addresses, private requests hide them. We ask how much work such a scheme must do, and which requests must do it, in the cell-probe model—with a passive server that only reads and overwrites addressed words. We examine the expected amortized per-request cost on randomized workloads with independent modes and uniformly random addresses. For array size n and client memory $m \le n^{1-\varepsilon}$ words, let $H = \log_2(2 + n/(m+1))$.

Our main result is a tight tradeoff between the cost of public reads and private writes. We show that if a request is private with probability $0 < p \le 1/2$ and private requests cost at most Q probes on average, then every scheme with $m \le n^{1-\varepsilon}$ has expected public-read cost

$$\Omega\left(\max\left\{1, \frac{H}{\log_2(2 + 1/p + Q/H)}\right\}\right),$$

even when the server learns the kind (read/write) of each private request. For polynomial m, we provide constructions that match this bound whenever Q exceeds a threshold that we identify for both Q treated as an expectation over the workload and as the average cost for the worst-case workload. In particular, at constant private frequency p a logarithmic private budget forces logarithmic public-read cost, and constant public-read cost requires private cost $(n/(m+1))^{\Omega(1)}$. This answers, for passive servers, the question posed by Zhang, Wang, and Weng, whose PIR-based scheme achieves constant public and logarithmic private bandwidth with server computation. Our lower bound proof fixes the public query and changes its private history; crucially, we charge them separately. Our constructions combine a public cache with linear merges of hash tables.

Two further groups of results concern public writes and total work. Firstly, we give a cost tradeoff between public writes and private reads. Secondly, for workloads mixing private operations, public reads, and public writes with probabilities p, r, a, we show that every scheme with $m \le n^{1-\varepsilon}$ has total cost per request $\Omega(\max\{1, H/\log_2(2 + 1/p)\})$ when private request kinds are hidden, and a construction with polynomial client memory matches this bound with a constant independent of p, r, a. We also separately bound the costs of reads and writes in this workload.

Lastly, public requests can in some cases be made cheaper by offloading their cost to preparation in earlier requests or cleanup in later ones. For both public reads and writes, we show localized tradeoffs constrained to a subset of requests, which demonstrate some of the limitations of such offloading.

Contents

1 Introduction **1**

1.1 Results 1 · 1.2 Techniques 2 · 1.3 Related work 4 · 1.4 Organization 4

2 Model **4**

2.1 Storage, probes, and traces 4 · 2.2 Leakage and privacy 4 · 2.3 Workloads, average costs, and budgets 5 · 2.4 Resources of the constructions 5

3 Main results **6**

3.1 The public-read frontier 6 · 3.2 Public writes 7 · 3.3 General workloads 8

4 Technical overview **9**

4.1 The lower bound 9 · 4.2 The matching construction 10 · 4.3 Public writes and general workloads 12

5 Open problems **13**

A Lower bounds for public reads **14**

A.1 Two information lemmas and the transfer conventions 14 · A.2 Proof of the per-epoch edge bound 16 · A.3 Quantitative details and private terminals 17 · A.4 Geometric epoch charging 18 · A.5 Proof of the public-read and private-request bounds 19

B The upper-bound construction, its schedules, and adaptive security **20**

B.1 Consuming tables and hidden permutations 21 · B.2 Removing duplicate versions without sorting 24 · B.3 Our hierarchies and the public and private access procedures 25 · B.4 Canonical export, resets, and cost allocation 30 · B.5 Adaptive requests with unchanged costs 32

C Budget inversion and precise tuning **35**

C.1 Smaller private average-input budgets 37

D Localized public-access charges and the read frontier **38**

D.1 A common accounting framework 38 · D.2 Localized public-read charge and its accounting bounds 39 · D.3 Lower bound for the localized public-read charge 40 · D.4 Matching constructions and the closing public request 42

E Public writes: lower bounds, constructions, and memory **43**

E.1 An average-cost tradeoff with disclosed private kinds 44 · E.2 Localized write charges and exact helper accounting 46 · E.3 Budget, cap, and three-class consequences 48 · E.4 Matched write regimes and their resources 50 · E.5 A calendar built from refined consuming tables 52 · E.6 Tuning the calendar 57 · E.7 Average write costs and a buffering example 58 · E.8 The high-private-budget localized-write frontier 60 · E.9 Bounded state on unbounded closing tails 64 · E.10 The client memory needed for inexpensive public writes 65

F General workloads: cost bounds and construction **69**

F.1 Average cost bounds on deterministic logical horizons 69 · F.2 Matching total cost for general workloads 73

G Computational parameters and encrypted storage **75**

H Variants of the read frontier **78**

H.1 Statistical trace privacy with counted routing 78

I Further comparisons and read-frontier examples **80**

I.1 Further related work 80 · I.2 Read-frontier examples 81

References **81**

AI methodology **85**

1 Introduction

An oblivious RAM (ORAM) lets a client store an array on an untrusted server without revealing which logical addresses it accesses [GO96]. The price is well understood when every access is private: with polynomially sublinear client memory, logarithmic overhead is necessary even for computational security [LN18, KL21]. Many computations, however, mix sensitive accesses with accesses whose addresses may be disclosed. Examples include ORAM initialization, batch loads, and statistical analysis. *Mixed-mode* ORAM, introduced by Zhang, Wang, and Weng [ZWW25], lets each request be marked public or private. How much work is necessary when only some accesses are private, and which requests must perform it?

These questions are non-trivial because the two modes interact. A public read of x must return the value of the latest private write to x without revealing which private request wrote it, and a public write must leave its value where a later private read can find it. Moreover, since the access pattern can depend on public requests, there are many ways to offload the work of public requests to others.

Zhang, Wang, and Weng obtain $O(1)$ public bandwidth and $O(\log_2 n)$ private bandwidth using communication-efficient private information retrieval (PIR) and sufficiently large blocks. Even their public reads use PIR, because a block's location can depend on earlier private evictions [ZWW25, Section 4.2], and they conjecture a logarithmic lower bound without it [ZWW25, Section 1.2]. We study a *passive cell-probe server* that only reads or overwrites an addressed word, whose trace reveals addresses, probe directions, and request boundaries, but neither cell contents nor returned answers.

1.1 Results

We analyze the expected amortized cost (further referred to as simply *average cost*) of ORAM requests, calculated over the randomness of the ORAM algorithm and the workload. In the randomized workloads, each request is independently public or private, addresses are independent uniform elements of $[n]$, and writes supply fresh uniform words. For array size n words, client memory m words, and probability of private access p, let

$$H = \log_2\Big(2 + \frac{n}{m+1}\Big), \quad F_p = \max\Big\{1, \frac{H}{\log_2(2+1/p)}\Big\}, \quad G_p(Q) = \max\Big\{1, \frac{H}{\log_2(2+1/p+Q/H)}\Big\}. \tag{1}$$

All logarithmic lower bounds assume $m \le n^{1-\varepsilon}$ for a fixed $\varepsilon > 0$, so that $H = \Theta(\log_2 n)$. Table 1 summarizes our results, Section 2 states the model we adopt and Section 3 presents the theorems.

Main result: a tight public-read/private-write tradeoff frontier. Consider workloads consisting of public reads mixed with private writes, with private probability $0 < p \le 1/2$. We show that if the average cost of a private access is Q probes, where Q exceeds a specified threshold, the optimal cost of a public read is $\Theta(G_p(Q))$ (see Theorems 3.1, 3.2 and Section 2.4). The lower bound holds even if request *kinds* (read/write) are disclosed. Three consequences illustrate the frontier (see Appendix I for more examples):

- At constant private frequency p, a logarithmic private budget forces logarithmic public-read cost. In particular, a passive server cannot match the constant-public, logarithmic-private guarantees of [ZWW25] (Corollary 3.3).
- If private requests are polynomially rare, $p = n^{-c}$, then $F_p = \Theta(1)$, so constant public-read cost is attainable at the natural private budget $Q = \Theta(H)$.

- At constant private frequency p, constant public-read cost requires private cost $(n/(m+1))^{\Omega(1)}$, and a private budget $(n/(m+1))^{\alpha}$ for any fixed $\alpha > 0$ suffices.

Our main technical novelty is separate accounting for work done by a terminal public read and work done earlier to move information to it. Combined with randomized workloads of IID requests, it allows us to obtain a tradeoff between public read and private write cost, with explicit dependency on the private request frequency p. Our new public cache and retiring merges give matching constructions that make the tradeoff tight. While the proof technique of fixing public queries and changing private updates already appears in the encrypted-multimap lower bounds of Patel, Persiano, and Yeo [PPY20], their logarithmic worst-case lower bound on the combined update and query cost does not preclude constant-time queries with logarithmic updates. Our bound forbids this at constant private frequency. When request kinds are hidden, the same frontier holds with private requests being reads. The matching constructions account for all client state, including the pseudorandom-function (PRF) workspace.

Public write/private read tradeoff. For public writes mixed with private reads, we give a tradeoff between the average cost of public writes and private reads (Theorem 3.4), permitting disclosed private request kinds. In particular, public write cost below a small multiple of F_p requires a private budget of order F_p/p, and with polynomial client memory that private budget suffices for inverse-polynomial public write cost (Corollary E.19).

Costs for general workloads with hidden private kinds. We show that if private operations, public reads, and public writes occur with probabilities p, r, a, and private request kinds are hidden, the average cost of all operations satisfies $T_N = p\Pi_N + rR_N + aW_N = \Omega(F_p)$, with a constant independent of p, r, a (Theorem 3.5). Moreover, a construction with polynomial client memory attains $T_N = O(F_p)$ (Theorem F.4). We also bound the average costs of public reads and public writes separately under budgets on the other requests (Theorems 3.6 and 3.7).

Localized public charges. The work of public requests may be partially performed by other requests via preparation or deferral. We demonstrate some of the limitations of such offloading by analyzing the tradeoff between private cost and localized read and write charges: for reads, the cost of *bridging reads*, whose addresses have not been accessed after the previous private request, together with their preceding public operations; for writes, the cost of *bridging writes*, whose addresses are not overwritten until the next private request, together with that request.

1.2 Techniques

Fixed public query with different private histories. Classical ORAM lower bound proofs compare executions that differ in the query [LN18, PY19, PY23], which a public query forbids. Thus, as in [PPY20], we fix the terminal public query and replace earlier private operations by fresh writes; an encoding argument then forces a large amount of information from older requests to be either transferred to younger ones or probed in the terminal read. Averaging over the inserted addresses preserves the terminal query's address distribution density up to a constant factor, so its work reflects the work of an average read. Keeping the transfer charge separate from the terminal charge then yields

$$R_N \log_2\left(2 + \frac{\Pi_N}{R_N} + \frac{1-p}{p}\right) = \Omega(H), \qquad R_N = \Omega(1), \tag{2}$$

Workload	Private request kinds for lower bound	Result	Upper-bound construction resources
Public reads with private writes	Disclosed	Average read cost $R_N = \Theta(G_p(Q))$ for average-input budget $Q \geq C_1 F_p$ and worst-case-input budget $Q \geq C_2 H$; same average read cost for the localized charge A_d (Theorems 3.1, 3.2, D.7)	$m \geq C \log_2^{12}(2+n)$ with $w = \Theta(\kappa^c)$, $c \geq 1$, $\kappa \leq n \leq \kappa^d$; or $m \geq n^\theta$ with $w = \Theta(\log_2 n)$
Public writes with private reads	Disclosed	A public write/private read tradeoff; private-budget threshold $\Theta(F_p/p)$ for small W_N (Theorem 3.4, Corollary E.19)	Polynomial workspace q, $q \geq Cp^{-2} \log_2 n$
Private requests, public reads, and public writes with probabilities p, r, a	Hidden	Average operation cost $T_N = \Theta(F_p)$; average read and write cost bounds for $r > 0$ and $a > 0$ respectively (Theorems 3.5, 3.6, 3.7, F.4)	Polynomial client workspace
Public writes with private reads	Disclosed (memory lower bound requires only correctness, no privacy)	Lower bounds for the localized charge A_w and client memory given an average/worst-case private budget with matching constructions in specified regimes (Theorems E.10, E.26)	Polynomial workspace in the matched regimes

Table 1: Results on IID uniform-address workloads; lower bounds, except for the correctness-only memory bound, assume $m \leq n^{1-\varepsilon}$ and $w \geq \lceil \log_2 n \rceil$. Constructions hide private request kinds, use $O(n)$ server cells, and account for PRF workspace (Section 2.4); initialization is charged separately.

where R_N is the average read cost, and Π_N is the average private cost. The term $(1-p)/p$ reflects that only private requests supply fresh information, even though both request kinds can contribute to information transfer.

Separating direct write costs from offloaded work. For public writes the roles are swapped: we fix public writes and vary the address of the final private read; the fresh values of the public writes supply entropy. The localized charge A_w ignores *helpers*, i.e. private requests that do not follow a public write. We account for this omitted work exactly, and we show how it relates to the average and worst-case input cost of private accesses as well as the memory lower bound.

A public cache and linear hash table rebuilding with deduplication. Our constructions are hierarchical and keep two structures. An *authoritative hierarchy* holds the current value of every address and serves private requests. In addition, each private request appends the record it accessed, with a fresh version, to an independently randomized *public index*. Both are built from *consuming hash tables* [AKL+20], in which a lookup removes the record it finds, so no table instance may be queried twice for the same key. Public reads additionally utilize a *public cache* indexed by disclosed addresses that records what earlier public reads found (including misses), and a repeated read of an unchanged address issues only dummy lookups to the hash tables. We rebuild hash tables

using *retiring merges*, which remove stale values. Our hierarchy has $L = \Theta(1 + \log_B(n/m))$ levels that grow by a factor of B, $O(L)$ cost per lookup, and $O(BL)$ amortized maintenance per append. Charging this maintenance to public or to private requests gives the $1/p$ and Q/H terms of the frontier.

1.3 Related work

Patel, Persiano, and Yeo [PPY20] prove logarithmic lower bounds for encrypted multi-maps, including plaintext-query settings; their public-query proof also changes private updates while fixing public queries, and Section 1.1 explains how our frequency-dependent tradeoff differs. We adapt the epoch and encoding framework of [LN18, PY19, PY23], which builds on the chronogram method [FS89] and information-transfer encodings [PD06]; Komargodski and Lin [KL21] give the general all-private memory/cell-size bound, of which we use only the regime of equal logical and physical word size.

Our constructions build on hierarchical rebuilding [GO96, KLO12], OptORAMa's consuming tables [AKL+20], PanORAMa's shuffled inputs [PPRY18], and linear deduplication [AKLS21]; the public cache, retiring merge, and write calendar are new. Write-public ORAM [AABD26] studies public writes with private reads; PIR-based mixed-mode ORAM [ZWW25], Onion ORAM [DvDF+16], and ORAM/PIR composition [AFN+17] use server computation. Appendix I provides further comparison, including with write-only ORAM [BMNO14], memory checking [BKV24], and the barriers of [WW18].

1.4 Organization

Sections 2–4 give the model, the main results, and a technical overview; complete proofs are in the appendices, by topic: public reads (A–D), public writes (E), general workloads (F), and conventions, variants, and comparisons (G–I). Generative AI tools assisted in developing the mathematics and in revising the exposition; the AI methodology statement at the end discusses the usage.

2 Model

2.1 Storage, probes, and traces

The logical array has n words of $w \geq \lceil \log_2 n \rceil$ bits. The client retains at most mw bits, counting all transient state. The passive server stores w-bit cells at addresses of $O(w)$ bits; a probe reads or overwrites one cell (an overwrite returns no old contents), and local computation is free. Request-independent preprocessing obeys the memory bound and is charged separately. Requests arrive online, each completing before the next arrives. Every read must be correct with probability at least $2/3$ on every fixed input (our constructions have negligible whole-execution error on each prescribed polynomial *horizon*, i.e. any polynomially bounded number of requests). The observed *trace* consists of probe addresses, probe directions, and logical request boundaries, but neither cell contents nor returned answers.

2.2 Leakage and privacy

The leakage $\mathsf{Leak}_{\mathrm{hid}}$ of an input request sequence is its mode sequence together with every complete public request, including addresses and write values; $\mathsf{Leak}_{\mathrm{dis}}$ additionally contains each private request's kind (read/write). Two inputs are *comparable* if their leakage and initial arrays agree: a hidden-kind comparison may change private addresses, values, and kinds; a disclosed-kind comparison only private addresses and values.

Statistical trace privacy on horizon $\bar{N}$ requires the traces of comparable inputs of length at most $\bar{N}$ to be within total variation distance $1/16$; it is *unrestricted* if this holds for all finite lengths. For statistical lower bounds on a deterministic horizon of N logical requests, privacy on horizon $N+2$ suffices. Lemma E.8, whose horizon instead counts macros of potentially unbounded logical length, assumes unrestricted statistical privacy. *Computational trace privacy* requires indistinguishability for polynomial-time samplers of comparable input pairs, with auxiliary information and polynomial probe cutoffs; samplers and distinguishers may be nonuniform (Appendix G gives the definition and states how the asymptotic bounds apply to any fixed implementation family). Computational lower bounds additionally require polynomial expected probes, including preprocessing, on the *exact reference workload* for which the theorem is stated. They require no execution-cost bound on the *altered workloads* used in the information-transfer proof and no cryptographic assumption. Our constructions satisfy the stronger *adaptive trace simulation* of Appendix B.5, against an environment choosing each request from earlier traces and logical answers, for arbitrary modes and the full read/write API.

2.3 Workloads, average costs, and budgets

We distinguish three *request classes*: private operations, public reads, and public writes, occurring with probabilities p, r, a, respectively, where $p + r + a = 1$ and $p > 0$. Request classes are IID and independent of the implementation's randomness. For a horizon of N logical requests, with N fixed in advance, define the *average costs*

$$\Pi_N = \frac{\mathbb{E}[\text{private probes}]}{pN}, \quad R_N = \frac{\mathbb{E}[\text{public-read probes}]}{rN}, \quad W_N = \frac{\mathbb{E}[\text{public-write probes}]}{aN}, \tag{3}$$

and $T_N = p\Pi_N + rR_N + aW_N$, the expected probes per request in the first N of them (an absent public class has average zero). The expectation is taken over the workload distribution and the algorithm's randomness. Every probe is charged to the request performing it, including maintenance done for other requests. Lower-bound workloads have independent uniform addresses and fresh uniform write values. The *two-class workloads* have private probability $0 < p \le 1/2$: $\mathcal{W}_{R,W}$ pairs public reads with private writes, $\mathcal{W}_{R,R}$ has public and private reads, and $\mathcal{W}_{W,R}$ pairs public writes with private reads.

A private *average-input budget* Q means $\Pi_N \le Q$ on the stated workload, while a *worst-case-input budget* P bounds the expected average number of probes per private request on every fixed input prefix by P, with expectation over the algorithm's randomness. Our constructions with a stated P-budget satisfy the stronger *pointwise* guarantee: on every input prefix and every random tape, the average number of probes per private request is at most P. An *individual cap* K bounds every private request on every random tape; neither of the two budgets implies a cap. In the computational setting, parameters, horizons, budgets, and $1/p$ are polynomially bounded, and p, r, a are efficiently sampleable.

2.4 Resources of the constructions

The read upper bounds require $m \ge C_0 \log_2^{12}(2+n)$ client words, $w \ge C \log_2 n$, and a security parameter κ with $\kappa \le n \le \kappa^d$ for a fixed d and polynomially bounded w. We assume an efficiently evaluable PRF [GGM86] secure against nonuniform polynomial-time distinguishers, and we *count* its retained keys and peak evaluation workspace within a reserved constant fraction of the mw client bits. For a conventional PRF, whose key and workspace take $O(\kappa^c)$ bits, two instantiations cover the matching claims (Table 7):

- *Polynomial client, logarithmic words*: $n^\theta \le m \le n^{1-\varepsilon}$, $w = \Theta(\log_2 n)$, $n = \kappa^d$ with $d > \max\{1, c/\theta\}$. Every computational matching result holds (write regimes with their stated lower bounds on m), with linear initialization.
- *Polylogarithmic client, large words*: $m = \lceil C_0 \log_2^{12}(2+n) \rceil$ and $w = \Theta(\kappa^c)$ for logical words and server cells alike. Only the read frontier is matched here, with $O(n \log_2 n)$ initialization.

Polylogarithmic memory with logarithmic words is covered by the lower bounds but not by our constructions, since a conventional key alone exceeds the client bits. Appendix G also gives a wrapper encrypting cell contents and discusses its overhead.

3 Main results

Throughout, $\varepsilon > 0$ is fixed, $m \le n^{1-\varepsilon}$, and H, F_p, G_p are as in (1). "The lower-bound conditions" are the storage, per-read correctness, privacy, and expected-probe qualifications of Section 2 for the theorem's reference workload; "the construction resources" are those of Section 2.4, including the PRF key and evaluation workspace.

3.1 The public-read frontier

Theorem 3.1 (Public-read lower bound)**.** *Fix $0 < p \le 1/2$ and a permitted horizon $N \ge 4n/p$. Consider $\mathcal{W}_{R,W}$ with disclosed or hidden private kinds, or $\mathcal{W}_{R,R}$ with hidden private kinds. Under the lower-bound conditions, with at most mw client bits, $w \ge \lceil \log_2 n \rceil$, and $O(w)$-bit physical addresses, every secure family with $\Pi_N \le Q$ satisfies $R_N \ge cG_p(Q)$; the same holds under a worst-case-input budget $P \le Q$. No cryptographic assumption or lower bound on m is needed.*

Theorem 3.2 (Matching computational constructions)**.** *Fix $0 < p \le 1/2$ and a permitted horizon $N \ge 4n/p$. Under the construction resources there are constants C_0, C, C_1, C_2 and families with hidden-kind adaptive computational security attaining*

$$\Pi_N \le Q, \quad R_N \le CG_p(Q) \qquad \text{for } Q \ge C_1 F_p, \tag{4}$$

$$P \le Q, \quad R_N \le CG_p(Q) \qquad \text{for } Q \ge C_2 H. \tag{5}$$

They use $O(n)$ server cells and count all retained and transient client memory. The bounds hold uniformly over requests chosen from earlier traces and logical answers, provided each mode is an independent Bernoulli draw and public requests are reads.

Theorem 3.1 is proved in Appendices A and C; the constructions, their adaptive security, and their tuning are in Appendices B and C. The smallest possible worst-case-input private budget for a secure full-API implementation is $\Theta(H)$. At the same time, the total cost on the specified workload can be as small as $\Theta(F_p)$ (Theorem C.3). Corollary 3.3 shows that the guarantees of [ZWW25] cannot be matched by a passive server.

Corollary 3.3 (No constant all-input public-read budget)**.** *Under the assumptions of Theorem 3.1 for a constant p, every secure implementation with average-input private budget $\Pi_N = O(H)$ satisfies $R_N = \Omega(H)$, even when private request kinds are disclosed. Consequently, such an implementation cannot bound the average public-read probes by a constant on every fixed prefix without public writes.*

Resources, smaller budgets, and variants. In the constructions, each public write costs $O(n \log_2 n)$ probes, and only $O_\theta(n)$ when $m \ge n^\theta$ (Corollary B.7). The constructions are adaptively secure (Corollary B.10).

Below the threshold C_1F_p, the picture depends on whether operation kinds are hidden. If they are, then $R_N \geq c_1 Q 2^{c_2 H/Q}$ for all $Q_0 \leq Q \leq c_0 F_p$, and this exponential order is attainable (Theorem C.4). If instead the kinds on $\mathcal{W}_{R,W}$ are disclosed, $R_N = O(F_p)$ is attainable with negligible average-input private cost thanks to offloading (Theorem C.5).

The frontier remains optimal if we consider the *localized charge* A_d, which counts each public read whose address has not been accessed after the previous private operation, together with this read's preceding public operation (Appendix D; see Section 3.2 for more about localized charges). The frontier's lower bounds also cover private read-modify-write calls, and with a larger client-bit allowance it can be matched by a statistically private construction on polynomial horizons (Appendix H).

3.2 Public writes

For the write bound, define the symmetric increasing function

$$\Phi(x, y) = \min\{x, y\} \log_2\left(2 + \frac{\max\{x, y\}}{\min\{x, y\}}\right) \quad (x, y > 0),$$

with value zero on the axes. It has the familiar algebraic form of symmetric query/update tradeoffs in cell-probe data structures [PD06].

Theorem 3.4 (Public write lower bound)**.** *Fix $0 < p \leq 1/2$ and $N \geq 4n$. Consider $\mathcal{W}_{W,R}$ with disclosed or hidden private kinds. Under the lower-bound conditions,*

$$T_N, \Pi_N = \Omega(1), \qquad \Phi(T_N, \Pi_N) = \Omega(H), \tag{6}$$

with constants independent of p. The result holds even for an implementation supporting only public writes and private reads.

Lemma E.1 proves the theorem. Note that it is a strengthening of Theorem 3.7 for $r = 0$. Its consequence is a private-budget threshold (Corollary E.19): $W_N \leq c_0 F_p$ forces $\Pi_N \geq c_1 F_p/p$, while with $m \geq \max\{n^\theta, Cp^{-2}\log_2 n\}$ a budget $\Pi_N = \Theta(F_p/p)$ permits $W_N = O(n^{-d})$. Thus, at constant p, logarithmic private cost permits inverse-polynomial write cost, whereas for reads it forces $R_N = \Omega(H)$.

Accounting for deferred work. Unlike a read, a write may buffer its update and leave substantial work to the next private query (Proposition E.17). To capture this deferred work, the *localized charge* A_w groups each run of public writes with its closing private query. It counts the last write to each address in the run and the closing query, then divides their expected total cost by the expected number of surviving writes. For example, two writes to distinct addresses followed by a private query contribute $c_1 + c_2 + c_{\text{priv}}$ to the cost and two to the write count. The measured prefix ends at the first private query after N; Appendix D gives the formal accounting.

This charge leaves out *helpers*: private requests with no new public writes since the preceding private request. They offer time for background maintenance, so making A_w small can move work onto them. After the first request, helpers occur with frequency p^2. An average private budget Q allows up to pQ helper work per request, but a cap of K probes on each private request allows only p^2K. Because of that, there is a factor $1/p$ difference between the optimal average budget and individual cap in our matched regimes.

Theorem E.9 gives optimal bounds on these two private costs and on client memory in a regime of polynomially rare private requests. For example, with $p = n^{-1/4}$, there exists a construction that

attains $A_w \le n^{-1/8}$ on every closed horizon using $\Theta(n^{1/2}\log_2 n)$ client words. Its private requests average $O(n^{1/4})$ probes and never exceed $O(n^{1/2})$; the memory, average budget, and cap are all optimal up to constants for $N \ge 4n$.

For every $0 < p \le 1/2$, we also lower-bound A_w under a private budget, and the private budget and individual cap required for a small A_w (Corollaries E.5 and E.7). A separate memory lower bound holds on every closed horizon and uses correctness alone (Theorem E.26).

The appendix gives the full parameter ranges and memory savings for shorter horizons. When public reads are also present, A_w includes the public reads preceding the selected writes and private operations (Lemma E.8).

3.3 General workloads

Theorem 3.5 (Total cost lower bound)**.** *Consider a workload with request-class probabilities p, r, a. Addresses are independent uniform elements of $[n]$, each private operation is a write independently with any fixed probability $\xi \in [0,1]$, and every write supplies a fresh uniform word. Under hidden private request kinds and the lower-bound conditions for this reference workload, for every permitted horizon $N \ge 4n/p$,*

$$T_N = aW_N + p\Pi_N + rR_N = \Omega\left(\max\left\{1, \frac{H}{\log_2(2+1/p)}\right\}\right). \tag{7}$$

The constant is independent of p, r, a, ξ; the probabilities must have polynomial descriptions and be efficiently sampleable in the computational case. The bound counts all probes on the first N requests, excluding initialization.

Theorem 3.6 (Public read lower bound with three request classes)**.** *Under the assumptions of Theorem 3.5, for any $r > 0$ (even without an inverse-polynomial lower bound),*

$$R_N = \Omega(1), \qquad R_N \log_2\left(2 + \frac{T_N}{pR_N}\right) = \Omega(H), \tag{8}$$

with constants independent of p, r, a, ξ.

Theorem 3.7 (Public write lower bound with three request classes)**.** *Under the assumptions of Theorem 3.5, for any $a > 0$ (even without an inverse-polynomial lower bound), with $u = a + p$,*

$$T_N/u, \Pi_N = \Omega(1), \qquad \Phi(T_N/u, \Pi_N) = \Omega(H). \tag{9}$$

Consequently, if $\Pi_N \le Q$ and $R_N \le V$, then

$$W_N \ge \frac{1}{a}\left[c\max\left\{F_p, \frac{uH}{\log_2(2+Q/H)}\right\} - pQ - rV\right]_+, \tag{10}$$

where $[z]_+ = \max\{z, 0\}$. The constants, including $c > 0$, are independent of p, r, a, ξ. An absent public-read class contributes zero. The same bound holds if $P \le Q$.

All three theorems are proved in Appendix F.1. Corollary F.3 gives the budget form of the read bound (analogous to (10)). Unlike R_N, W_N has no unconditional constant lower bound—a buffering implementation makes W_N inverse-polynomially small even with all three request classes present (Proposition E.17).

The total-cost bound is tight with polynomial client memory: for $n^\theta \le m \le n^{1-\varepsilon}$ and the construction resources, Theorem F.4 gives an adaptively secure full-API implementation with $O(n)$

server words, public requests costing $O(F_p)$, and private budget $O(F_p/p)$ on any input prefix, so $T_N = O(F_p)$ with a constant independent of r, a.

4 Technical overview

We briefly present the core of the lower and upper bound results comprising the read frontier (full proofs: Appendices A, B, and C), then describe the changes for public writes and general workloads.

4.1 The lower bound

The mechanism. A public read making Y probes inspects at most Y cells. Consider a *terminal* public read and an *epoch* of ℓ consecutive requests that happened before it, possibly with some *younger* requests in between. The writes from the epoch carry about $p\ell$ fresh words, far more than the client's m words once $\ell \gg m/p$. For the terminal read to return a uniformly chosen one of them without probing a cell written in that epoch, the information would need to have been carried forward by younger requests that read $\Omega(p\ell)$ cells written in the epoch. Requests of either mode can do this, at total rate of T_N. Thus, each epoch is either touched by the terminal read or costs the younger requests $\Omega(p\ell)$ reads from it. Now consider adjacent epochs of geometric lengths $\ell_0\rho^i$, where $\ell_0 \approx m/p$, lying in a window of length $\approx n/p$. There are about $H/\log_2\rho$ such epochs. They are in a decreasing order, and the shortest epoch is adjacent to the terminal read. The terminal read touches at most Y epochs. Meanwhile, averaging over the terminal's position, any given read probe can be charged only $O(1/\rho)$, where probing an older epoch of length ℓ corresponds to charge $1/\ell$. Summing gives total charge $Y + (T_N/p)/\rho = \Omega(H/\log_2\rho)$, and $\rho = \Theta(2 + T_N/(pY))$ gives $Y\log_2(2 + T_N/(pY)) = \Omega(H)$. The term $T_N/(pY)$ represents the work available per fresh private word relative to the terminal read's work. The remainder of this subsection formalizes each step of the argument. The workload alterations we utilize refine those of [PPY20, full version, Section 5.2, Theorem 19] in the two ways described in Section 1.2.

The reference experiment. Fix an endpoint range, window length, and shortest epoch length

$$\mathcal{V} = \{\lceil N/2\rceil, \ldots, N-1\}, \qquad \omega = \left\lfloor \frac{n}{128p} \right\rfloor, \qquad \ell_0 = \left\lceil \frac{C(m+1)}{p} \right\rceil.$$

Choose an endpoint e uniformly in $\mathcal{V}$ and consider a *reference prefix* of length e drawn from the measured workload. The *safe set* V_e consists of all addresses not written in the last ω requests of this prefix. On $\mathcal{W}_{R,R}$, $V_e = [n]$, and on $\mathcal{W}_{R,W}$ at most $p\omega \le n/128$ addresses are excluded in expectation, so the event $G_0 = \{|V_e| \ge 15n/16\}$ has probability $g \ge 7/8$. Now draw a reference prefix conditioned on G_0 and form a *reference input* by appending a terminal public read of an address uniform in V_e. Its conditional address density is at most $16/15$ relative to a uniform query and the conditioned prefix has relative density at most $1/g$, so the expected probe count Y of this read satisfies $Y \le 4R_N$.

A *dependency edge* joins two consecutive physical probes to the same cell (potentially with probes to other cells in between) when the second is a read. For an epoch I of length $\ell \ge \ell_0$ inside the window, at a fixed position relative to e, let E_I count the edges from I to the *younger* requests, and D_I the edges from I to the terminal request; both are observable from the trace.

Lemma 4.1 (Per-epoch edge bound)**.** *Under the assumptions of Theorem 3.1, with $m + 1 \le C_{\mathrm{mem}} n^{1-\varepsilon}$, there are constants $C, c, \eta > 0$, independent of p, such that for all sufficiently large*

parameters and every epoch I as above,

$$\mathbb{E}_{\mathrm{ref}} D_I + \frac{\mathbb{E}_{\mathrm{ref}} E_I}{\eta p \ell} \geq c. \tag{11}$$

The expectations are over the reference terminal experiment: its endpoint, prefix, terminal address, and implementation randomness; they are conditioned only on G_0. The lemma permits disclosed kinds on $\mathcal{W}_{R,W}$ and requires hidden kinds on $\mathcal{W}_{R,R}$.

The altered experiment. Let G be the event that a given epoch has at least $p\ell/2$ private positions. This event has constant probability conditioned on G_0. Draw a reference prefix conditioned on $G_0 \cap G$. Form an *altered* prefix by replacing the first $k = \lfloor p\ell/4 \rfloor$ private positions of the epoch by writes of independent words $X_1, \ldots, X_k$ to a uniformly random tuple of distinct addresses $A_1, \ldots, A_k \in V_e$. Next, choose a uniform $J \in [k]$ and append a public read of A_J to both prefixes. Its output for the altered prefix is X_J. Averaging over the addresses $A_1, \ldots, A_k$ and J, the distribution of the reference input generated this way is identical to the reference experiment conditioned on G.

The proof of Lemma 4.1 has two steps. The *information step* looks only at the altered execution. Before J is chosen, an encoder sends the client state after the epoch and, for each edge from it to a younger request, the contents of its cell. The encoder aborts if there are more than $a_I = \lfloor \eta p \ell \rfloor$ such edges. From this message, a decoder can replay the younger requests exactly and, if $D_I = 0$, also the terminal read, which returns X_J. However, the encoder's message has only at most $kw/16$ bits, too few to encode the k fresh words (Lemma A.2). Hence, this procedure must often fail; specifically, the event $\{E_I > a_I \vee D_I > 0\}$ has constant probability. Next, the *privacy step* shows that because this event can be efficiently checked from the trace and the epoch description alone, its probability on the reference execution changes only by a small constant (Lemma A.3). Removing the conditioning on G and applying Markov's inequality gives (11).

Geometric epochs and per-edge charging. The epochs follow the chronogram method [FS89], and we adapt the randomized-endpoint charge of [PY23, full version, Lemma 4]. For large n, $m \leq n^{1-\varepsilon}$ means that $\omega \geq 2\ell_0$ and

$$h_* := \log_2(2 + \omega/\ell_0) = \Theta(H). \tag{12}$$

The shortest epoch has no younger requests, so $E_I = 0$ and Lemma 4.1 gives $Y \geq c$. For an integer $\rho \geq 2$, there are $\Omega(h_*/\log_2 \rho)$ disjoint epochs I_i of lengths $\ell_i = \ell_0 \rho^i$ before the terminal. They have at most Y edges to the terminal. By Lemma A.5, each edge of the prefix adds only $O(1/\rho)$ to the normalized charge $\sum_i E_{I_i}/\ell_i$ summed over all endpoints. Averaging over endpoints gives $\sum_i \mathbb{E} E_{I_i}/\ell_i = O(T_N/\rho)$. With $X = T_N/p$, summing (11) gives $C_1 Y + C_2 X/\rho \geq c_1 h_*/\log_2 \rho$, and $\rho = \lceil 2 + X/Y \rceil$ yields

$$Y \log_2(2 + X/Y) \geq c h_*. \tag{13}$$

Then, monotonicity in Y, $Y \leq 4R_N$, and $X = \Pi_N + rR_N/p$ give (2). Making the terminal read private gives a companion bound on Π_N under hidden request kinds (Theorem A.1). Lemma A.6 packages this reasoning for reuse with more general experiments in which fresh words occur at density d, and adds a symmetric form $\Phi(X, Y) = \Omega(h_*)$ used for public writes and general workloads.

4.2 The matching construction

Both the authoritative hierarchy and the public index have a client buffer of $q = \Theta(m)$ slots. An integer growth factor $2 \leq B \leq \lceil n/q \rceil$ gives hierarchy depth $L = \lceil \log_B(n/q) \rceil$. Each level holds one

consuming hash table. When a level overflows, its surviving records are merged into a *batch*, which is *carried* to the next level. A level overflows on every B-th received batch. A lookup costs $O(L)$, and an append costs $O(BL)$ amortized.

The hierarchical structures and the public cache. A private request searches the authoritative hierarchy from youngest to oldest level, with its real key until the first hit and dummies thereafter, then appends its current value with a fresh version (even on a read). This append ensures that the same key is not queried twice in the same table instance. The same record is appended to the index, which has independent randomness. The construction also maintains a server cache C, where $C[x] = (v_x, t_x, \text{last}(x))$ stores the value, version, and completion time of the latest public read of x, and every hash table instance stores a public creation time. A public read of x reads $C[x]$ and the client buffer after performing its assigned maintenance, then queries each active (non-empty) table T in the index with the real key when $\text{last}(x) < \text{creation}(T)$ and with a dummy otherwise, keeps the largest version found, and updates $C[x]$. It appends no record to either hierarchy. The index allows public reads to consume their own copies of data while leaving the authoritative hierarchy for private requests.

For example, if a carry places a private write $(x, v, 7)$ in a new table instance U, the next public read of x queries U and caches $(v, 7)$. Later public reads of x use dummies in U until a private write $(x, v', 8)$ moves from the buffer to a new table instance. Recording misses similarly prevents repeated real queries.

Linear rebuilds with a polylogarithmic client. We use OptORAMa's consuming tables [AKL+20, Functionality 4.7 and Theorem 8.8], in which a successful lookup removes its record, extraction returns the remaining records in a secret random order, builds are linear on secretly shuffled inputs, and lookups are constant-probe. Linear deduplication of independently shuffled inputs is established in [AKLS21, Section 2.4 and Theorem 4.1], but in our case querying an old table to delete duplicates could repeat an earlier public lookup. We avoid this problem and maintain independent shuffles via the following *retiring merge* of a padded, secretly shuffled array A of distinct newer records with an older source table T: (1) retire T and extract its remaining records into O; (2) build a temporary consuming table U from O with fresh hashes; (3) scan all slots of A, querying U once for each real key and issuing a dummy query for each dummy slot, ignoring the answers; (4) extract U and obliviously intersperse it with A, which retains the newer version of every key. Each step is linear in the public padded lengths. The set extracted from U depends on A only through its set of keys, and extraction shuffles it independently of A's order (Lemma B.4). Now, since a carry merges a shuffled batch from the top-level buffer with several consecutive hash tables ordered from youngest to oldest, we can implement it by repeatedly performing retiring merges, then normalizing to the destination's capacity by adding/removing dummies. Geometric level sizes make the whole procedure linear in that capacity.

Maintenance allocation. There are several hidden-kind *schedules* that allocate rebuilding caused by private operations differently:

Schedule	Private cost bound	Public-read cost bound
I: perform all carries immediately when due	$P = O(H + BL)$	$R_N = O(L)$
G: defer index carries to public reads	$P = O(H)$	$R_N = O(L + pBL)$
A: try to defer all carries	$\Pi_N = O(L)$	$R_N = O(L + pBL)$

Schedules I and G have authority growth factor 2. Deferral uses $q+\Delta$ buffer slots, $\Delta = \lceil \log_2^2(2+n) \rceil$. Overflow of the authoritative buffer forces an $O(n)$ private carry, whereas overflow of the public index buffer invalidates the index and forces an $O(n \log_2 n)$ public rebuild of the cache. An overflow happens with negligible probability under IID modes with $p \le 1/2$; under other mode sequences it costs more but leaks nothing, since its timing is mode-determined. A disclosed-write variant of schedule A, can skip the authority lookup (schedule W in Appendix C.1).

The frontier: combining the lower bounds and constructions. On either reference workload, (2) and $\Pi_N \le Q$ give $R_N \log_2(2 + 1/p + Q/R_N) \ge cH$, which in turn gives $R_N = \Omega(G_p(Q))$ (Lemma C.1); the same also follows from a worst-case-input budget P. Substituting $R_N \le T_N/r$ gives $T_N = \Omega(F_p)$, and routing every request through the private interface forces $P = \Omega(H)$ (Theorem C.3).

To match the total-cost lower bound $T_N = \Omega(F_p)$, choose $B = \Theta(\min\{1/p, \lceil n/q \rceil\})$. Then $pB = O(1)$ and $L, pH = O(F_p)$, so schedule A gives $\Pi_N, R_N, T_N = O(F_p)$, and schedule G gives $R_N, T_N = O(F_p), P = O(H)$. For $Q \ge CH$, schedule I permits $B = \Theta(\min\{Q/H, \lceil n/q \rceil\})$ with constant small enough that $P \le Q$, and gives $R_N = O(\max\{1, H/\log_2(2 + Q/H)\})$; choosing the cheapest of the three schedules gives $R_N = O(G_p(Q))$. This proves (5) and the high-budget part of (4). For $CF_p \le Q < CH$ schedule A suffices, since $G_p(Q) = \Theta(F_p)$ (see Appendix C for the full proof).

4.3 Public writes and general workloads

The public write lower bound (6) uses the same epoch framework with the roles reversed (as in the standard chronogram technique [PY19, PY23]): the terminal is a *private* read whose address may differ between the altered and reference inputs, all public requests stay fixed, and the entropy comes from the fresh values of the epoch's public writes. This allows disclosed request kinds. The lower bound follows from the symmetric form of Lemma A.6 (see Lemma E.1). For the localized charge A_w, each public write is selected with probability $p/(p + (1-p)/n)$. Using this, Lemma E.3 relates A_w to the regular costs and helper work. Combining this with another non-localized tradeoff (Lemma E.1) gives bounds on A_w depending on the private budget and cap.

The matching construction uses a *write calendar* to schedule background rebuilds with fixed deadlines, using copies of table records that lookups leave unchanged. Credits for rebuilding accrue on every private request but are spent only by helpers (Lemma E.12), which yields a worst-case input private budget $O(L/p)$ together with an individual cap $O(L/p^2)$. A public request completes a rebuild only if its deadline would otherwise be missed, with probability $e^{-\Omega(p^2 q)}$. Table 6 summarizes the write results and their proofs.

With three request classes, the lower bound splits by the rate of potential writes $u = a + p$ (with hidden kinds, all private operations can be replaced by writes). If $u \ge 1/4$ (which includes $p > 1/2$ and $r = 0$), a private terminal and Lemma A.6 give $\Phi(T_N/u, \Pi_N) = \Omega(H)$, hence $T_N = \Omega(uH/\log_2(2 + u/p))$ (Lemma F.1); otherwise the fixed-public-query proof applies with a safe set also excluding the window's public writes, and there is a constant floor $R_N = \Omega(1)$ (Lemma F.2). Theorem F.4 matches the resulting total-cost lower bound. The private-terminal tradeoff holds for every u; inverting it under $\Pi_N \le Q$ and subtracting the private and public-read work from T_N gives Theorem 3.7.

5 Open problems

Total cost and the read frontier are tight in their stated resource regimes, while the write results have matching constructions for specified budgets, latencies, and memories. Open questions include:

- *Write frontiers and three request classes.* Characterize the write cost at every private budget, close the gap in the localized-write-charge frontier, and give matching constructions for the joint average cost bounds for three-class workloads at separate budgets. Reduce the write and three-class total-cost constructions to polylogarithmic client memory.
- *Read resources and latency.* Give matching constructions for the frontier with constant client memory and logarithmic words. Extend the frontier to larger cells, as in the all-private bound [KL21], resolve the frontier gap at $Q = \Theta(F_p)$ with hidden kinds, and obtain a public-read cap equal to the average budget under the same private budget (ordinary ORAM admits worst-case logarithmic overhead [AKLS21]).
- *Workloads and leakage.* Determine the frontier when public queries operate on a small subset of addresses, its complete dependence on the horizon N, and its form when public-read answers are fully disclosed. Give statistically secure matching constructions without requiring the overhead on the client memory from the key placement of Appendix H (optimal statistical all-private ORAM is known for large transfer blocks [KS26]).

A Lower bounds for public reads

This appendix proves Theorem A.1, which contains the tradeoff (2) behind Theorem 3.1 together with a companion bound on the private cost. It follows the outline of Section 4.1 and uses its notation. Throughout, $H = \log_2(2 + n/(m+1))$ and $T_N = p\Pi_N + (1-p)R_N$.

Theorem A.1 (Tradeoff lower bounds for public reads and private requests)**.** *Suppose that logical words and server cells have w bits, physical addresses have at most λw bits for a fixed λ, and $w \geq \lceil \log_2 n \rceil$. All retained and temporary client information occupies at most mw bits. For fixed $\varepsilon, C_{\mathrm{mem}} > 0$, assume $m + 1 \leq C_{\mathrm{mem}} n^{1-\varepsilon}$. Let $0 < p \leq 1/2$ and $N \geq 4n/p$. Consider the workload $\mathcal{W}_{R,W}$, on which private request kinds may be disclosed, or the workload $\mathcal{W}_{R,R}$, on which they must be hidden (Section 2.3). An online implementation with per-read correctness at least $2/3$ on every fixed full-API input and trace privacy as defined in Section 2 satisfies, for all sufficiently large n,*

$$R_N \geq c, \qquad R_N \log_2\left(2 + \frac{\Pi_N}{R_N} + \frac{1-p}{p}\right) \geq cH, \tag{14}$$

where $c > 0$ depends only on the fixed constants. On either workload with hidden kinds, also

$$\Pi_N \geq c, \qquad \Pi_N \log_2\left(3 + \frac{(1-p)R_N}{p\Pi_N}\right) \geq cH. \tag{15}$$

The statistical statement needs only statistical privacy on horizon $N + 2$, that is, total-variation distance $1/16$ on comparable pairs of length at most $N + 2$; every experiment below ends by position $N+1$. For computational privacy, we additionally require that the parameters and N are polynomial in the security parameter, that p is efficiently samplable, and that the exact reference execution (including observed preprocessing) has polynomial expected trace length. The inequalities then hold eventually for every fixed secure implementation family.

Section A.1 states the two information lemmas and the conventions used by every lower bound in the paper, Section A.2 proves the per-epoch edge bound (Lemma 4.1), Section A.3 collects the quantitative details and the private-terminal variant, and Section A.5 completes both tradeoff bounds.

Initial contents and request-independent preprocessing may be arbitrary, subject to the memory bound; the decoders below receive the actual initial contents and preprocessing outcome of the run as side information Z (as in Lemma A.2), which is independent of the fresh source words. Without loss of generality, the preprocessing and each request have their own independent random tapes, so a simulation can resume after a skipped epoch without storing a tape offset. We also assume $w \geq 4$, as the theorem is asymptotic in n.

A.1 Two information lemmas and the transfer conventions

We write $H(X \mid Z)$ for conditional entropy and $I(X; V \mid Z)$ for conditional mutual information, with base two logarithms. Thus $I(X; V \mid Z) = H(X \mid Z) - H(X \mid V, Z)$.

The first lemma is a standard random-access encoding consequence of Fano's inequality [CT06, Section 2.10].

Lemma A.2 (A message sent before the queried coordinate)**.** *Let $X_1, \ldots, X_k$ be independent uniform w-bit words, independent of side information Z. An encoder sends a fixed-length β-bit message before an independent uniform $J \in [k]$ is supplied. A decoder given the message, Z, and J recovers X_J with probability s only if*

$$s \leq \frac{\beta}{kw} + \frac{1}{w}.$$

Consequently $\beta \le kw/16$ *and* $w \ge 4$ *imply* $s \le 5/16$.

Proof. Let V be the message. Independence and entropy subadditivity give $\sum_{j=1}^{k} I(X_j; V \mid Z) \le I(X_1, \dots, X_k; V \mid Z) \le \beta$. Since V is chosen before J, $I(X_J; V \mid Z, J) \le \beta/k$. Fano's inequality, or its elementary proof by first encoding the error indicator and then the incorrectly predicted word, gives $H(X_J \mid V, Z, J) \le 1 + (1-s)w$. Subtracting from $H(X_J \mid Z, J) = w$ proves the claim. □

Lemma A.3 (Transfer of an observable event)**.** *Let* $(U_{\mathrm{alt}}, U_{\mathrm{ref}}, Z)$ *be a jointly sampled pair of pointwise comparable inputs and auxiliary information, independent of the implementation's random tapes, and let* E *depend only on* Z *and the address trace. Statistical privacy with distance* δ *implies* $\Pr_{\mathrm{ref}}(E) \ge \Pr_{\mathrm{alt}}(E) - \delta$. *For computational privacy suppose that*

(T1) the input pair is samplable by a nonuniform polynomial-time sampler and its length and fields are polynomial;

(T2) E *can be tested in time polynomial in the observed trace length;*

(T3) the exact reference trace has expected length at most a polynomial $P(\kappa)$.

Then $\Pr_{\mathrm{ref}}(E) \ge \Pr_{\mathrm{alt}}(E) - 1/64 - \nu(\kappa)$ *for a negligible function* ν.

Proof. The statistical claim follows by conditioning on the chosen pair and averaging. Computationally, choose the polynomial cutoff $b = \lceil 64(P+1) \rceil$. A distinguisher accepts if the trace overflows b probes, or if it completes and E occurs. Its acceptance on the altered input is at least $\Pr_{\mathrm{alt}}(E)$, whereas its reference acceptance is at most $\Pr_{\mathrm{ref}}(E) + P/b$. Truncated-trace indistinguishability proves the claim; negligible sampling error only changes this bounded test negligibly. □

Neither an expected-cost bound on the altered input nor an efficient decoder is needed in Lemma A.3. Every lower bound in this paper applies it in the same way, so we fix the following conventions once.

Convention A.4 (Verifying the transfer hypotheses)**.** In each lower-bound experiment of this paper:

(a) *Samplers.* Altered/reference samplers use only requests and independent coins. Conditioning events of constant probability are sampled by rejection, and distinct address tuples in polynomial expected time; polynomial caps give negligible total-variation error. Since the horizon is polynomial and $N \ge 4n/p$, $1/p$ is polynomial. This gives (T1).

(b) *Tests.* The observable event is tested by tracking the latest probe to each cell, without simulating the encoder or decoder. This gives (T2).

(c) *Reference expectation.* Forcing the class of one terminal request multiplies a reference expectation by at most $1/p$ (or $1/(ap)$, $1/r$, as stated where used); conditioning on constant-probability events and restricting the query to a safe set multiply it by only constants. The exact-reference expected-probe qualification of Section 2 therefore gives (T3). The altered input may have a much larger expected cost.

(d) *Decoder termination.* Each entropy decoder simulates the altered execution from known server memory, overwriting (*repairing*) the cells listed in the encoder's message with their supplied contents. A cell is *unrepaired* if the simulation reads it while it holds stale contents not supplied by the message. The *exact-reconstruction event* is the event that no such read occurs, so the simulation reproduces the altered execution exactly. Outside this event, a simulation with unrepaired cells need not terminate. Executions on the altered input terminate, so there is a finite cap on their local steps exceeded with probability at most $1/64$. The cap is chosen before

sampling the source words, tapes, and coordinate, and the decoder's simulation is truncated at it, returning 0^w on abort or timeout. On the exact-reconstruction event this loses at most $1/64$ success probability. The cap may depend on the implementation and public parameters, need not be polynomial, and is used only by the entropy decoder, never by the privacy sampler or the observable-event test. It is separate from the probe cutoff of Lemma A.3.

(e) *Advice and uniformity.* Nonuniform advice selects the epoch scale ρ from the expected costs; no efficient procedure for computing expected costs is needed. Each proof applies Lemma A.3 for several choices of endpoint and epoch, and we need a single negligible ν valid for all of them. Call the excess acceptance loss $\Pr_{\mathrm{alt}}(E) - 1/64 - \Pr_{\mathrm{ref}}(E)$ the *transfer gap*. If for infinitely many parameter settings some choice had a nonnegligible transfer gap, advice could encode that choice for each setting, yielding one nonuniform distinguisher that breaks truncated-trace indistinguishability. Hence one negligible ν covers all the polynomially bounded choices used.

A.2 Proof of the per-epoch edge bound

We use the reference experiment, epoch I, and edge counts of Section 4.1, and the altered prefix of Section 4.1. Subscripts and superscripts alt and ref refer to the altered and reference executions, respectively. The two lemmas above separate the message bound, which concerns the altered execution only, from the privacy transfer, which moves an observable event to the reference execution.

Proof of Lemma 4.1. Write $\mu = p\ell$. Let G be the event that I has at least $\mu/2$ private positions; a binomial tail bound gives $\Pr(G \mid G_0) \geq 1 - e^{-\mu/8}/g \geq 63/64$. Conditional on $G_0 \cap G$, select the first $k = \lfloor \mu/4 \rfloor$ private positions in I, a uniformly random ordered tuple of distinct addresses $A_1, \ldots, A_k \in V_e$, and independent uniform w-bit words $X_1, \ldots, X_k$; the tuple exists because $k \leq p\omega/4 \leq n/512 < |V_e|$. Replace the selected requests by private writes of X_j to A_j. After forming this altered prefix, choose an independent uniform $J \in [k]$ and append the public read of A_J, forming the altered input. All inserted writes are distinct and all unchanged writes in the window avoid V_e, so the correct answer on the altered input is X_J.

The paired reference input appends the same public read to the original private prefix instead (Section 4.1). The inputs are pointwise comparable: on $\mathcal{W}_{R,W}$ all changed positions remain private writes; on $\mathcal{W}_{R,R}$ changing their kinds uses hidden-kind privacy. Conditional on any reference prefix, averaging the tuple and J makes A_J uniform on V_e, so the reference marginal distribution is exactly the terminal experiment of Section 4.1 with the additional conditioning on G. The altered input need not follow the IID workload, since correctness and privacy hold on every fixed comparable pair, and only reference execution costs are charged to the workload.

Let $a_I = \lfloor \eta\mu \rfloor$. The entropy decoder's side information Z_{dec} consists of the reference prefix, the address tuple, the common pre-epoch client and server state, and all implementation tapes; it is independent of the source words and J. The encoder sees the altered execution up to and including e and sends its message *before* J is drawn; only then does the decoder receive J and attempt to recover X_J. Separately, the privacy test receives only the trace and a sampler-generated epoch description and threshold, independent of the implementation's coins; it receives no internal states or tapes.

The message contains an abort bit, the post-epoch client state, and a_I padded address/value records. The encoder aborts if $E_I^{\mathrm{alt}} > a_I$; otherwise it sends the post-epoch contents of the cells on those edges. For sufficiently small η and large C,

$$\beta \leq 1 + mw + O(a_I w) \leq kw/16;$$

Section A.3 gives the constants. Online execution makes the entire message independent of J, including the repair list determined by the younger requests.

On no abort, the decoder installs the supplied client state and cell records in the known pre-epoch server memory and simulates the younger requests. There is at most one edge counted by E_I^{alt} per cell—it ends at that cell's first younger probe. If this probe is a read, the supplied record repairs the cell; if it is a write, no old contents are needed. Cells untouched in I already have their correct contents. Induction over probes makes the simulation exact. Once J is revealed, the terminal simulation is also exact if $D_I^{\mathrm{alt}} = 0$. With the finite simulation cap of Convention A.4(d), averaging per-read correctness over the sampled fixed inputs recovers X_J except with probability at most $1/3 + 1/64 + \Pr_{\mathrm{alt}}(E_I^{\mathrm{alt}} > a_I \vee D_I^{\mathrm{alt}} > 0 \mid G_0, G)$.

By Lemma A.2, a β-bit message sent before an independent uniform coordinate recovers it with probability at most $\beta/(kw) + 1/w \le 5/16$. Hence, in the altered execution, the event $\{E_I^{\mathrm{alt}} > a_I \vee D_I^{\mathrm{alt}} > 0\}$ has probability at least $65/192 > 1/3$. Its test needs only the epoch description and the observed address trace with operation boundaries, so Lemma A.3 transfers it to the reference event $\{E_I^{\mathrm{ref}} > a_I \vee D_I^{\mathrm{ref}} > 0\}$ with only constant loss; Convention A.4 verifies the computational hypotheses, with the forced public terminal costing a factor at most $1/p$ since the public mode has probability at least $1/2$. Multiplying by $\Pr(G \mid G_0)$ gives a constant lower bound on the reference event conditioned only on G_0. Finally,

$$\Pr_{\mathrm{ref}}(E_I^{\mathrm{ref}} > a_I \vee D_I^{\mathrm{ref}} > 0 \mid G_0) \le \frac{\mathbb{E}_{\mathrm{ref}} E_I^{\mathrm{ref}}}{a_I + 1} + \mathbb{E}_{\mathrm{ref}} D_I^{\mathrm{ref}}, \qquad a_I + 1 > \eta p \ell,$$

which proves the claim. □

A.3 Quantitative details and private terminals

Window and endpoint. For sufficiently large n, $\omega \ge 2\ell_0$ and every endpoint in $\mathcal{V}$ has a preceding window of length ω. Moreover $\omega/\ell_0 = \Theta(n/(m+1))$, so the fixed polynomial client memory gap gives (12). The probability $g = \Pr(G_0)$ is independent of the endpoint e, since the reference fields in a length-ω window are identically distributed at each endpoint.

Encoding constants. If physical addresses have at most λw bits, take $A_{\mathrm{adr}} = \lceil \lambda \rceil + 2$ so that a validity bit, an address, and a cell value fit in $A_{\mathrm{adr}} w$ bits. With $a_I = \lfloor \eta p \ell \rfloor$ and $k = \lfloor p\ell/4 \rfloor$, the padded message has length at most $1 + mw + a_I A_{\mathrm{adr}} w \le kw/16$ for sufficiently small η and sufficiently large C. These constants depend only on the fixed address-size constant. The final application of Markov's inequality uses expectations conditioned on G_0. Conditioning on the private-count event G was used to construct the comparable pair, then removed by multiplying its constant conditional probability. The event in the altered execution therefore has probability at least $2/3 - 1/64 - 5/16 = 65/192$, as used above.

Terminal cost and the private variant. For a public terminal, let Y have the meaning of Section 4.1. Under hidden-kind privacy we also use a private reference terminal at an address uniform on V_e: a read on $\mathcal{W}_{R,R}$, or a write of an independent uniform word U on $\mathcal{W}_{R,W}$. Again condition the reference prefix only on G_0 and let Y be the expected terminal cost in this variant. The address density is at most $16/15$ relative to a uniform query. Since the mode and address of the next request in the workload are independent of the prefix, in both cases

$$Y \le \frac{16}{15g} \frac{N}{|\mathcal{V}|} Z \le 4Z, \qquad Z \in \{R_N, \Pi_N\}, \tag{16}$$

and the prefix distribution has density at most $1/g$ relative to the unconditioned reference prefix.

To extend (11) to the private terminal, use the same altered-prefix coupling and encoding as in Lemma 4.1. The altered terminal is a private read of A_J. On $\mathcal{W}_{R,R}$ the reference terminal is the same read; on $\mathcal{W}_{R,W}$ it is a private write of U to A_J, a change of kind permitted by hidden-kind privacy. Include U in the decoder's side information. The altered execution still answers X_J, and the averaged reference address is still uniform on V_e. The same proof therefore gives (11) for this private-terminal experiment.

A.4 Geometric epoch charging

Every lower bound in this paper converts a per-epoch edge bound of the form (11) into a tradeoff by summing it over geometrically arranged epochs. We isolate this step as Lemma A.6, so that each application (public or private terminals, public-write sources, three request classes) need only specify its entropy density, window, an upper bound on terminal cost in terms of workload cost, and constant lower bounds on the relevant costs. For an endpoint e and integer $\rho \geq 2$, the epochs are $I_i = [e - s_i - \ell_i + 1, e - s_i]$, and their younger request spans are $[e - s_i + 1, e]$, where $\ell_i = \ell_0 \rho^i$ and $s_i = (\ell_i - \ell_0)/(\rho - 1)$. This arrangement follows the chronogram framework [FS89], and the randomized-endpoint charge is adapted from [PY23, full version, Lemma 4].

Lemma A.5 (Per-edge charge)**.** *Fix a dependency edge of the reference execution between requests $v_1 < v_2$, and put $z = v_2 - v_1$. For each epoch I_i, there are at most $\min\{z, \ell_i, s_i\}$ endpoints such that the edge starts in I_i and ends in its younger span. Moreover, such endpoints exist only if $z \leq \ell_i + s_i$. Consequently*

$$\sum_{i:\, z \leq \ell_i + s_i} \frac{\min\{z, \ell_i, s_i\}}{\ell_i} = O(1/\rho). \tag{17}$$

Proof. The bounds on e follow by intersecting the endpoint intervals for which $v_1 \in I_i$ and $v_2 \in [e - s_i + 1, e]$. These two conditions constrain e to intervals of length at most ℓ_i and s_i, whose intersection has length at most z. For (17), we split the epochs by size, assuming $\rho \geq 2$.

- *Small epochs, $\ell_i < z/2$.* Since $s_i \leq \ell_i/(\rho - 1) \leq \ell_i$, we have $\ell_i + s_i \leq 2\ell_i < z$, so these epochs contribute nothing.
- *Middle epochs, $z/2 \leq \ell_i \leq \rho z$.* At most three epochs lie in this range, and each contributes at most $s_i/\ell_i \leq 1/(\rho - 1)$.
- *Large epochs, $\ell_i > \rho z$.* Each contributes at most z/ℓ_i, and these terms form a geometric series with first term below $1/\rho$, so they sum to $O(1/\rho)$.

□

Lemma A.6 (Epoch charging)**.** *Fix a prefix length N and let T_N be the average total cost of the first N requests. Fix also a window length ω and shortest epoch length ℓ_0 with $\omega \geq 2\ell_0$, and put $h_* = \log_2(2 + \omega/\ell_0)$. Consider a reference experiment that draws an endpoint $e \geq \omega$ uniformly from a set $\mathcal{V}$ of at least $N/4$ positions, draws the reference prefix of length e conditioned on an event G_0 whose probability is at least a fixed $g > 0$, and appends a terminal block of at most two requests with expected probe count Y. For an epoch I in the window at a fixed position relative to e, let E_I and D_I count the dependency edges from I into the younger requests and into the terminal block. Suppose that for a density of fresh words $d > 0$ and constants $c, \eta > 0$, every epoch I of length $\ell \geq \ell_0$ in the window satisfies*

$$\mathbb{E} D_I + \frac{\mathbb{E} E_I}{\eta d \ell} \geq c, \tag{18}$$

with the expectations being over the conditioned experiment. Put $X = T_N/d$. Then, with constants depending only on c, η, g:

(i) $Y \geq c$;

(ii) $Y \log_2(2 + X/Y) \geq c_1 h_*$;

(iii) if additionally $X \geq c_0$ for a fixed $c_0 > 0$, then $\Phi(X, Y) \geq c_2 h_*$, with c_2 also depending on c_0.

Proof. (i) The epoch of length ℓ_0 placed immediately before the terminal has no younger requests, so $E_I = 0$ and (18) gives $\mathbb{E}D_I \geq c$. Each terminal physical read has at most one incoming edge, so $Y \geq \mathbb{E}D_I$.

(ii) Consider geometric epochs with an integer growth factor $\rho \geq 2$. The number of such epochs in the window is $k_\rho = \lfloor \log_\rho(1+(\rho-1)\omega/\ell_0) \rfloor = \Omega(h_*/\log_2 \rho)$. Note that this holds even for $\rho > \omega/\ell_0$, since the first epoch always fits. Each terminal physical read has at most one incoming edge, whose earlier endpoint lies in at most one of the disjoint epochs, so $\sum_i \mathbb{E}D_{I_i} \leq Y$. Every dependency edge within the reference prefix ends at one of its read operations, so the prefix has at most NT_N expected edges. Lemma A.5 charges a total normalized weight $O(1/\rho)$ to every edge, across all epochs and endpoints. Averaging over the at least $N/4$ endpoints, including a factor $1/g$ from the conditioning, we get

$$\sum_i \frac{\mathbb{E}E_{I_i}}{\ell_i} = O(T_N/\rho). \tag{19}$$

Summing (18) over the k_ρ epochs gives

$$C_1 Y + C_2 X/\rho \geq c' h_*/\log_2 \rho. \tag{20}$$

Choosing $\rho = \lceil 2 + X/Y \rceil$, so that $X/\rho \leq Y$, yields

$$Y \log_2(2 + X/Y) \geq c_1 h_*. \tag{21}$$

(iii) The symmetric form needs a slightly different epoch arrangement. We adapt the first epoch construction of Persiano and Yeo [PY19, full version, Section 4.1]. For an integer $b \geq 2$, there is not one but $b-1$ consecutive epochs of each size $\ell_i = \ell_0 b^i$. The spans of younger requests have lengths $s_{i,j} = (j+1)\ell_i - \ell_0$ for $0 \leq j \leq b-2$. For $2 \leq b \leq \sqrt{1+\omega/\ell_0}$ there are $\Omega(bh_*/\log_2 b)$ full disjoint epochs. An edge spanning z operations can start in an epoch of length ℓ_i and end in its younger span only if $\ell_i \geq z/b$, and all epochs of this size contribute total normalized weight at most $(b-1)\min\{1, z/\ell_i\}$. At most two sizes lie between z/b and z, and larger sizes form a geometric tail, giving $O(b)$ total weight per edge. Averaging endpoints, summing (18), and dividing by b yields

$$C_3 X + C_4 Y/b \geq c'' h_*/\log_2 b. \tag{22}$$

Now if $X \geq Y$, choose $\rho = \lceil 2 + X/Y \rceil$ in (20) to obtain $\Phi(X, Y) = Y \log_2(2 + X/Y) = \Omega(h_*)$. If $X < Y$, choose $b = \lceil 2 + Y/X \rceil$ in (22); when b lies in its permitted range, $Y/b \leq X$ gives $\Phi(X, Y) = X \log_2(2 + Y/X) = \Omega(h_*)$, and when it exceeds that range, the bounds $\log_2(2 + Y/X) \geq \frac{1}{2}h_* - O(1)$ and $X \geq c_0$ give the same conclusion. □

Computationally, X is polynomially bounded and $Y \geq c$, so $\rho = \lceil 2 + X/Y \rceil$ and b in its permitted range have polynomial-length descriptions and may be supplied as advice (Convention A.4(e)).

A.5 Proof of the public-read and private-request bounds

Proof of Theorem A.1. Apply Lemma A.6 with density $d = p$, the window and shortest epoch of Section 4.1, the event G_0, and the public or private terminal. Lemma 4.1 and its private-terminal variant supply (18), and (12) gives $h_* = \Theta(H)$. Parts (i) and (ii) give $Y \geq c$ and $Y \log_2(2 + T_N/(pY)) \geq cH$.

The function $y \log_2(2 + x/y)$ is increasing in $y > 0$ and changes by at most a constant factor under constant scaling of y. Together with (16) this gives $Z \geq c$ and $Z \log_2(2 + T_N/(pZ)) \geq cH$ for $Z = R_N$ or $Z = \Pi_N$. Substituting $T_N/p = \Pi_N + (1-p)R_N/p$ proves both inequalities in Theorem A.1; the private-terminal alternative requires hidden-kind privacy on either workload. Since $p \leq 1/2$, $T_N/p \geq R_N$, so $T_N/p = \Omega(1)$ as well. □

Remark A.7 (Disclosed kinds on the read workload)**.** Under the construction resources of Theorem 3.2, there is a full-API implementation with disclosed-kind trace privacy that achieves $\Pi_N = O(H)$ and $R_N = 1$ on $\mathcal{W}_{R,R}$, using $O(n)$ server cells and the same asymptotic client memory. Thus, the hidden-kind restriction for the read workload is necessary, even at constant p. The construction is as follows.

Use schedule G from Theorem B.1 limited to private requests as an authoritative backend, alongside a plain server array holding a public mirror. Initialize both consistently, with initialization charged separately. A private read uses the backend, while a public read probes the corresponding mirror cell. A private write updates the backend, then rebuilds the entire mirror by reading backend addresses $1, \ldots, n$ in order and writing each returned value to its fixed mirror cell. A public write updates the backend and the fixed mirror cell.

The mirror is up to date at every request boundary. Its rebuild schedule depends only on operation kinds, which are disclosed. Moreover, the rebuild's probe locations are fixed, and backend privacy hides all private addresses and values. Hence, the construction's trace has disclosed-kind privacy for the full API. The extra backend scan uses only constant client workspace, accommodated by the usual constant-factor memory allocation. The backend horizon is chosen in advance to cover all internal calls, so its negligible-error guarantee continues to apply. On $\mathcal{W}_{R,R}$ no write or mirror rebuild occurs: each private request makes one backend call, whose worst-case-input prefix-amortized cost is $O(H)$, and each public request uses exactly one probe. Expensive rebuilds are charged to private writes on other inputs.

B The upper-bound construction, its schedules, and adaptive security

We give the construction underlying all the upper bounds, following the outline of Section 4.2. Below, the *authority* is the authoritative hierarchy, the *index* is the public index, and a *table instance* is one consuming hash table, created by a build and retired by its extraction. The consuming tables require secretly shuffled build inputs. The retiring merge utilizes consuming-table lookups to remove old versions of keys in linear time. Exporting the logical array into a plain one, without modifying the server data structures (*canonical export*), costs $O(n \log_2 n)$ probes with a polylogarithmic client; Corollary B.7 reduces this to $O(n)$ with polynomial client workspace without changing the measured frontiers. Section B.1 states the imported primitives and the shuffle invariant they require, Section B.2 proves the merge, Section B.3 specifies the hierarchy with its procedures and proves the correctness of the outputs and consuming hash table lookups, Section B.4 treats exports and resets and proves Theorem B.1, and Section B.5 proves adaptive security with unchanged costs.

Fix $\varepsilon > 0$, $\kappa \leq n \leq \kappa^d$ for fixed d, $w \geq C \log_2 n$, and a workspace parameter

$$\log_2^{12}(2 + n) \leq q \leq n^{1-\varepsilon}.$$

Assume the shared PRF key and its entire evaluation workspace fit in $O(qw)$ bits, reserved as part of the client memory. All other parameter, execution-length, and computational assumptions are

those of Section 2. Put

$$H_q = \log_2(2 + n/q), \qquad 2 \le B \le \lceil n/q \rceil, \qquad L = \lceil \log_B(n/q) \rceil, \qquad \Delta = \lceil \log_2^2(2 + n) \rceil.$$

Each record, including its address, value, version, and other metadata, occupies $O(1)$ words. Constants may depend on ε and on the fixed exponents and coefficients of the polynomial parameter bounds, but not on p, B, Q, or the horizon N.

Theorem B.1 (Small-client maintenance schedules)**.** *There are full-API implementations using $O(q)$ total client words and $O(n)$ server words with adaptive computational trace simulation and negligible whole-execution correctness error. The hidden-kind schedules* I *(perform all carries immediately),* G *(defer private index carries to public reads), and* A *(defer all regular private carries to public reads) satisfy*

schedule	*private cost bound*	*public-read cost bound*
I	$P = O(H_q + BL)$	$R_N = O(L)$
G	$P = O(H_q)$	$R_N = O(L + (p/(1-p))BL) + \mathrm{negl}(\kappa)$
A	$\Pi_N = O(L) + \mathrm{negl}(\kappa)$	$R_N = O(L + (p/(1-p))BL) + \mathrm{negl}(\kappa)$.

The average-input bounds hold at every a-priori chosen polynomial horizon under fresh IID modes with $p \le 1/2$, no public writes, and arbitrary logical requests chosen adaptively from earlier traces and logical responses. For these constructions, the private P-bounds additionally hold pointwise: on every full-API input and random tape, a prefix with I private requests incurs at most PI private probes, without charging initialization to them. The disclosed-kind schedule W *(Appendix B.3) supports the full API and, on the workload with every private request being a write, satisfies*

$$\Pi_N \le Cnp^{\Delta-1}, \qquad R_N = O(L + (p/(1-p))BL) + \mathrm{negl}(\kappa).$$

Initialization and each public write cost $O(n \log_2 n)$ probes.

Table 2 separates imported guarantees from the adaptations and caller invariants proved here. Linear deduplication itself has precedent in [AKLS21]; the merge below handles retirement and joint shuffling for our consuming interfaces. Canonical export reads outside the consuming-table API, so we separately prove that it preserves privacy jointly with subsequent table operations. The adaptive-security theorem uses this additional guarantee.

B.1 Consuming tables and hidden permutations

The use of randomly shuffled inputs and residual randomness in hierarchical ORAM originates in PanORAMa [PPRY18]; the concrete primitives used below are from OptORAMa [AKL+20].

A padded array is *secretly shuffled* if, in the ideal execution, its order is a uniform permutation conditional on its multiset of records and the public transcript. Independent shuffles mean independent permutations conditional on the logical multisets, which may themselves be correlated or overlap. Identical dummy entries can be given distinct conceptual labels when defining these permutations; the implementation need not expose or use such labels.

Joint shuffle invariant. Write $\mathcal{L}(X \mid Y)$ for the conditional distribution of X given Y, and $\mathsf{Unif}(S)$ for the uniform distribution on a finite set S. Consider the ideal-primitive execution. At a merge entry or an interface that consumes shuffled arrays (build, interspersing, or normalization), let $\mathcal{H}$ comprise the application's prior requests, responses, auxiliary information, and simulated public history, and let $\mathcal{M}$ comprise the unordered logical records of all live components and pending

Component	Imported	Proved in this paper	
		Adaptation or new part	Caller obligation
Consuming table (Lemma B.2)	Reactive consuming table [AKL+20].	Keep final stashes on the client; hide probe directions; charge the shared PRF evaluator.	Distinct real keys; shuffled build input; each real key, even a miss, looked up at most once.
Array primitives (Section B.1)	Compaction and interspersing [AKL+20].	Normalize padded arrays without revealing how many entries are real.	Inputs satisfy (23); real records fit the destination capacity.
Merge and carries (Lemmas B.4, B.5)	Linear deduplication [AKLS21].	Rebuild the older residual under fresh hashes; delete overlaps in one full scan of the newer array; intersperse.	Newer versions win; sources are retired; (23) holds at the next shuffled input.
Public cache (Section B.3)	Defined in this paper	Creation and completion timestamps decide which queries are real; cache the largest version any component returns.	No real query repeated per table instance; the cache dominates every record a completed public lookup consumed.
Export and simulation (Lemma B.6, Theorem B.9)	Oblivious sorting; [GM11] in Corollary B.7.	Export by fixed scans and sorting; compose the lifetime table simulators with a separate argument for exports.	Export keeps only canonical logical data and leaves live state unchanged; schedules and capacities depend only on leakage.

Table 2: Ingredients of the upper bound. The second column lists what is imported; the adaptations, new parts, and caller obligations in the last two columns are proved in this paper. The sorter of [GM11] is used only for initialization and canonical export in Corollary B.7.

arrays. If $A_1, \ldots, A_k$ are the arrays whose hidden orders are used at this interface, with public lengths $a_1, \ldots, a_k$, require

$$\mathcal{L}(\pi_{A_1}, \ldots, \pi_{A_k} \mid \mathcal{H}, \mathcal{M}) = \bigotimes_{j=1}^{k} \mathsf{Unif}(\mathcal{S}_{a_j}). \tag{23}$$

Dummy slots are conceptually labeled, and subsequent primitive coins are fresh and independent. Lemma B.5 proves that the hierarchy maintains (23).

Lemma B.2 (Consuming-table primitive)**.** *For a public capacity $q \le s = O(n)$, a secretly shuffled padded array of distinct real keys has a consuming-table implementation with $O(s)$ server space, an $O(\log_2 \kappa)$-record client stash, and the following operations.*

1. Build *costs $O(s)$ probes.*
2. Lookup(x) *costs $O(1)$ probes, returns the record for x or a miss, and consumes a found record. Every real key, including an absent key, may be queried at most once during the table instance's lifetime; polynomially many dummy queries are allowed.*
3. Extract *retires the instance and returns its unconsumed records, padded to length s, in a fresh secret random order, using $O(s)$ probes.*

These operations reactively realize the corresponding ideal functionality, up to negligible error, for admissible nonrecurrent calls, including calls selected from earlier traces and logical outputs. Their leakage consists of the public capacities and operation sequence; real/dummy choices, hits/misses, and unconsumed counts are hidden. All instances use and account for a single PRF evaluator, and each instance tags its inputs to the PRF with a unique identifier. The additional local working storage, excluding that evaluator and the consuming table stashes, fits in $O(q)$ words. Probe and storage bounds can be made pointwise (i.e. a cap on every operation), including after failure (Convention B.3).

Proof. We instantiate the consuming-table primitive using CombHT, the combined oblivious hash-table construction of OptORAMa [AKL+20, Construction 8.7], with the following adaptations.

Imported interface and costs. Use the OptORAMa construction [AKL+20, Definition 3.3, Functionality 4.7, Construction 8.7, and Theorem 8.8], with its stateful online simulator. The November 18, 2020 version incorporates the hierarchical stash-handling correction prompted by Alibi [FNO21]. For $s \geq \log_2^{11} \kappa$ its build and extraction costs are $O(s + s \log_2 s / \log_2^2 \kappa) = O(s)$, since $s = \mathrm{poly}(\kappa)$. Its lookup has constant server work besides scans of two stashes of $O(\log_2 \kappa)$ records. The construction's small-bin stash arrays are already combined into a server table, so only the two final stashes (OFS and CombSS in OptORAMa's notation) require local retention. The primitive fails with probability at most $s^2 \exp[-\Omega(\log_2 \kappa \log_2 \log_2 \kappa)]$ in the random-function hybrid, in addition to PRF distinguishing advantage. Instance and subtable identifiers give disjoint domains of the shared PRF. The condition $q \geq \log_2^{12}(2 + n)$ implies satisfying the imported size threshold for all sufficiently large parameters, with slack for constant-factor allocations, and the word-size prerequisites follow from $w \geq C \log_2 n$.

Probe-model adaptations. Retain both final stashes in counted client memory, replacing accesses to their fixed padded regions by local operations. If a build creates them on the server, copy each complete region in a fixed scan and thereafter use only the local copy. These transfers cost $O(\log_2 \kappa)$ per build or extraction, and local scans cost no probes. The remaining server trace is a public projection of the imported trace. To hide probe directions, expand every remaining primitive cell access at address a into a read of a followed by a write of a: for an original read, write back the unchanged value; for an original write, discard the read value and write the intended contents. The expanded directed trace is a deterministic transformation of the imported address trace, with the cost changing at most by a constant factor and identical state at the end of each original access. In particular, deleting a matching record and leaving a nonmatching record unchanged both have the same read/write pattern.

Workspace placement. OptORAMa's RAM model has $O(1)$ private CPU words; accesses to its other working memory are part of its observed trace [AKL+20, Section 3.1]. We place that memory on the server: the main payload arrays, the combined small-bin stash table, hash-assignment metadata, and build/extraction scratch. Their accesses are already covered by the imported probe bound, and they do not contribute any unaccounted client workspace. Our only persistent local additions for a table are its two final stashes and a constant number of control words. The CPU registers and the single PRF evaluation workspace are reused across sequential calls. The hierarchy's public counters and pending caller state occupy $O(\log_2 n)$ further words in total; Table 4 accounts for these together with the buffers and all live stashes.

Reactive contract and caller obligations. Retain one simulator for the complete lifetime of each table. Substituting the simulator preserves the caller's auxiliary state, functional outputs, and observed trace jointly through subsequent interaction; the simulator receives only capacities and commands. The input assumptions use OptORAMa's randomized-input convention [AKL+20, Section 3.1]. Admissible callers must preserve nonrecurrence (not querying a consumed real key)

and the joint shuffle condition (23). Section B.3 proves nonrecurrence, Lemma B.4 checks the merge-specific dependence, and Lemma B.5 proves (23) for the whole hierarchy; Theorem B.9 then constructs the online simulator. Together with the bounded continuation of Convention B.3, reactive composition and the PRF hybrid prove the claim. □

Convention B.3 (Bounded continuation after a primitive failure)**.** Every construction in this paper uses the following resource discipline. Use the public linear probe bound of each build and extraction, fixed bounds for each lookup, preallocated scratch arrays, and a polynomial local-step cap. An exceptional path is stopped at its bound and replaced by a padded dummy state of the declared capacity; out-of-range or malformed physical addresses are mapped to a fixed scratch cell. Subsequent calls retain the scheduled capacities and counters and never retry a failed construction. This means that no failure triggers an additional carry, reset, or private-paid recovery, so prefix and space bounds hold on every random tape. Uniform integer samples used by the shuffle primitives are generated locally with capped rejection sampling, $\lceil \log_2^2(2+n) \rceil$ trials per sample sufficing for negligible total sampling error without server probes. In the ideal-correct execution these caps change the construction's behavior only with negligible probability. While correctness and privacy may no longer hold after a failure, its probability is negligible over the whole polynomial execution, so a polynomial union bound together with the shared-PRF hybrid gives negligible total correctness and privacy error.

We also use deterministic linear-probe tight compaction and the linear-probe interspersing and real/dummy interspersing algorithms of [AKL+20, Theorem 5.1, Claims 6.3 and 6.7, and Algorithm 6.6]; for tight compaction see also the earlier standalone treatment of Peserico [Pes18]. Interspersing two independently secretly shuffled padded arrays of lengths a, b returns a secretly shuffled array of length $a+b$ in $O(a+b)$ probes. Tight compaction is indiscriminate: its permutation depends only on the real/dummy marks, so it does not preserve the original order of real records. A secretly shuffled array can be *normalized* to any public capacity c known to contain its real population in $O(a+c)$ probes by compacting, truncating or appending dummies to length c, and applying real/dummy interspersing. The number of real records remains hidden. These operations use $O(a+c)$ server space and fit within the same local workspace allowance.

B.2 Removing duplicate versions without sorting

Asharov, Komargodski, Lin, and Shi [AKLS21, Theorem 4.1] give linear oblivious deduplication for independently shuffled arrays, retaining the preferred copy. The lemma below is the *retiring merge* of Section 4.2. It adapts their procedure by retiring the old table and building a fresh temporary table, while ensuring that no real key is queried more than once in a table instance and that (23) is preserved.

Lemma B.4 (A linear merge with newer records taking precedence)**.** *Let A be a secretly shuffled padded array of length a with distinct real keys, and let T be an older consuming table of public capacity $s \geq q$. Assume that each record of A is newer than the record with the same key in T, if one exists, and that (23) holds at merge entry with T's unconsumed records T_{live} included in $\mathcal{M}$. There is an admissible procedure using $O(a+s)$ probes and server scratch words that retires T and returns a secretly shuffled, length-$(a+s)$ padded array with real records*

$$A_{\text{real}} \ \cup \ \{z \in T_{\text{live}} : \text{key}(z) \notin \text{keys}(A_{\text{real}})\}.$$

The output has distinct real keys. The procedure reveals neither the overlap nor the number of real records and preserves the joint shuffle invariant needed for subsequent merges and builds.

Proof. Run the following fixed sequence of actions, with all arrays padded to their public lengths.

M1. Retire T and compute $O \leftarrow T.\mathsf{Extract}()$. No later operation accesses that instance through its lookup or extraction interface.

M2. Build a temporary table $U \leftarrow \mathsf{Build}(O)$ with a fresh PRF domain and public capacity s.

M3. For each of the a slots of A, in order, read its record into a constant-size register. Query its key in U if real, and issue a dummy query otherwise. Discard the answer and leave A unchanged.

M4. Compute $D \leftarrow U.\mathsf{Extract}()$, retiring U.

M5. Obliviously intersperse A and D, obtaining a padded array of length $a + s$.

All real queries in M3 are distinct because the real keys of A are distinct, including queries that miss in U. At the end of M3 exactly the keys also present in A have been consumed from O, so M4 returns the stipulated older residual and M5 has the claimed logical output. The two extractions, temporary build, scan and lookups, and interspersing together cost $O(a + s)$.

The ideal joint distribution. Use $\mathcal{H}$ and $\mathcal{M}$ from (23), evaluated at merge entry, and work in the ideal-table execution under this conditioning. M1 gives the older live set a fresh permutation, independent of A's order, so M2 has a valid shuffled input. M3 makes exactly a lookup calls; its query *order* depends on A's permutation, but its final residual set depends only on keys(A_{real}). Let $\widehat{\tau}_{1:4}$ be the simulated trace of M1–M4. The table simulators see only capacities and commands, so this trace does not expose A's order, and M4 gives the older residual a fresh independent permutation. With conceptually labeled dummy slots,

$$\mathcal{L}(\pi_A, \pi_D \mid \mathcal{H}, \mathcal{M}, \widehat{\tau}_{1:4}) = \mathsf{Unif}(\mathcal{S}_a) \otimes \mathsf{Unif}(\mathcal{S}_s),$$

where π_A, π_D are the permutations of the corresponding padded multisets and $\mathcal{S}_t$ is the set of permutations of t slots. M5 therefore has the independent shuffled inputs it requires, and its ideal output is a fresh hidden uniform permutation of their union, jointly with the logical records of the remaining components. Note that (23) is not guaranteed midway through M3: there, a partial residual can reveal a prefix of A and bias its conditional order. However, this is harmless, since no shuffled input is consumed until after the full scan and extraction, by which point the residual depends only on the complete key set.

Applying the reactive contract. In M3 the caller holds the ordered newer array A and obtains successive keys by scanning it. Conditional on the logical multisets, this array is independent of U's fresh build randomness, so it is admissible auxiliary state in Lemma B.2. The fresh U is essential: querying T could repeat an earlier public lookup, and extracting a copy while continuing to query the original would not implement a retiring extraction. □

B.3 Our hierarchies and the public and private access procedures

Each hierarchy has a client buffer of $q + \Delta$ slots, capacities $S_i = qB^i$ for $1 \le i < L$ and $S_L = n$, and digits $d_i \in \{0, \dots, B-1\}$ for $i < L$. A nonzero digit makes the respective level active; the bottom has its own active flag. Active levels are consuming tables. Buffer occupancy counts appended requests rather than distinct keys: redundant or unused records remain padded slots. A record consists of a logical address, its value, and a version. A shared global version counter increases on every private request and public write, and a separate monotone event clock orders table creations and completed public queries. All counters and identifiers have $O(\log_2 n)$ bits on the prescribed (a-priori chosen) polynomial horizon and are retained across resets.

Initially the global version counter is one and the authority's bottom contains the input array at that version; its other levels and buffer are empty. The index starts empty, with the input array at version one in its public cache and sentinel completion times. Within either hierarchy, records in a younger component have newer versions than records of the same address in an older component, since carries move records only toward older levels, every append receives a fresh version, and consumption only deletes records.

Operation order and schedule policies. Table 3 specifies the only schedule choices used by Procedures 1–4. A structure not listed as deferred carries immediately when an append fills its q regular slots. A deferred authority carries on a public read, or on the private append that fills all $q + \Delta$ slots; a deferred index instead becomes dirty at that limit. Occupancy counts slots, including duplicates and dummies. All procedures are sequential, and no request arrives during maintenance. In Procedure 4 and the carry analysis, B, L, S_i refer to the hierarchy being serviced.

Schedule	Authority growth factor	Index growth factor	Deferred structures
I: private index work	2	B	None
G: defer index	2	B	Index
A: defer all carries	B	B	Authority and index
W: disclosed writes	B	B	Authority and index

Table 3: Maintenance policies. Public writes reset both structures. Schedule W skips the authority search for a disclosed private write.

Procedure 1 (Private access to x)**.**

1. Unless this is a disclosed write in schedule W, search the authority buffer for its newest record for x, then query every active authority table from youngest to oldest. Use x until the first hit (including a buffer hit), and dummies thereafter. Save the read answer if applicable.
2. Advance the global version counter. Append the supplied write value or saved read value, with x and this version, to the authority buffer and, if clean, to the index buffer. A dirty index receives no append.
3. Service the authority first: if its occupancy reaches q under an immediate policy or $q + \Delta$ under a deferred policy, call Procedure 4. Next service the index if it's clean: if its occupancy reaches q under an immediate policy, call Procedure 4; if it reaches $q + \Delta$ under a deferred policy, discard the index's tables and buffer (this uses no server probes), mark it dirty, and suspend index appends.
4. Return the saved read answer or a write acknowledgment.

Procedure 2 (Public read of x)**.**

1. Complete a due deferred authority carry if there is one. If the index is dirty, export the authority without changing the original, replace the public cache by this current array with sentinel completion times, and reset the index to clean with empty buffer and levels. Otherwise complete a due deferred index carry if there is one. This finishes all maintenance before the query phase.
2. Read $C[x]$ and save its old completion time τ. Collect its cached record and the newest buffered record for x, if any. Query every active index table T, using x when $\tau <$ creation(T), and a dummy otherwise. Add every returned real record to the candidates and continue through all tables, including after a hit.
3. Choose the largest version among the collected records. Advance the event clock and write this record and the new completion time to $C[x]$. Return its value. Note that no new record is appended to the hierarchy.

Procedure 3 (Public write of v to x)**.**

1. Export the authority, including its buffered records, without first requiring a carry.
2. Advance the version counter and apply the disclosed write of v to address x in the canonical (logical address order) export output, using the new version.
3. Reset both hierarchies from the updated output as in Lemma B.6, retaining the global version, event, and instance counters.

   All export and reset work is charged to that public write.

Procedure 4 (Carry in a hierarchy with growth factor B)**.** The buffer has b occupied slots, where $q \le b \le q + \Delta$.

1. Let $d_j, j < L$ be the first digit below $B-1$, or set $j = L$ if there is none. Deduplicate the entire buffer locally, retaining the latest version of each key and marking its older slots dummy.
2. Remove the oldest q slots from the buffer, including dummies. Independently shuffle this batch locally and write it in a fixed scan to a padded array A. Leave the remaining $b - q \le \Delta$ slots in the buffer.
3. Merge A with each active (*source*) level among levels $1, \ldots, j$ in increasing order, using Lemma B.4. This retires the merged levels.
4. Normalize the result to public length S_j and build a fresh destination table instance under a fresh unique identifier. Advance the event clock and assign the table's creation time after the build completes. Clear digits below j; increment d_j if $j < L$, and otherwise clear all digits and mark the bottom table active. The carry is now complete.

Carry capacity and work. Since $\Delta < q$, one carry suffices at each maintenance opportunity. The procedures describe the ideal-correct path; the bounded continuation of Convention B.3 supplies padded defaults on exceptional paths without changing the public schedule.

At each merge the accumulated newer records have distinct keys, and chronology ensures the record version precedence requirement, so the final output contains the newest surviving version at each address. The digit population invariant is that level $i < L$ has at most $d_i q B^{i-1}$ real records. A carry into $j < L$ adds at most $q + \sum_{i<j}(B-1)qB^{i-1} = qB^{j-1}$ records to a destination whose digit was at most $B-2$; the new population is at most $(B-1)qB^{j-1} \le S_j$, and duplicate removal and consumption can only reduce it. At the bottom level, distinct keys give a population bound n.

Let $s_1, \ldots, s_k$ be the scanned source capacities. They form a geometrically increasing subsequence, except that the last, clipped bottom capacity may be within a factor two of its predecessor. The padded array length before merge i is $q + \sum_{h<i} s_h$, so the total merge work is at most

$$C \sum_{i=1}^{k} \left(q + \sum_{h \le i} s_h \right) = O(S_j).$$

This is because the geometric sum stopping at any nonbottom level is at most twice its capacity, the sum of these capacities is $O(S_j)$, and the possible extra bottom term is also $O(S_j)$. The same estimate holds for $k = 0$, when only normalization and the new build are needed, which also cost $O(S_j)$. After F appends, at most $\lfloor F/q \rfloor$ batches have been flushed from the buffer into the hierarchy, and a carry reaches level i on every B^{i-1}-th flushed batch; deferral only changes who pays its cost. The carry cost bound on an input prefix is consequently

$$\sum_{i=1}^{L} CS_i \left\lfloor \frac{F}{qB^{i-1}} \right\rfloor \le CFBL, \tag{24}$$

since $S_i/(qB^{i-1}) = B$ for $i < L$ and is at most B at the bottom. Discarding or resetting an index partitions its appends into epochs, and summing the same bound over these epochs creates no additive initialization term. Every individual carry costs $O(n)$ probes. These bounds include all temporary tables, their lookups, normalization, and builds.

Authority correctness and nonrecurrence. The buffer and tables are searched from youngest to oldest, using a real key until the first hit and dummies thereafter. A read returns and appends the found value to the buffer; a write appends its supplied value. Both kinds perform the same search and assign a fresh version. A consumed authority record is replaced by that append before the operation returns, so the current value of every address remains represented, and older surviving copies are superseded. A real query to x in a table instance, including a miss, is protected by the appended record in the buffer or a younger level until that instance is retired; local buffer deduplication cannot remove this protection, since it removes a record only in favor of a still newer buffered record. Thus no table instance receives a repeated real key query. For disclosed private writes the lookup may be skipped; the fresh append still preserves correctness, chronology, and this nonrecurrence argument.

Index correctness and cache domination. We keep the public cache $C[x] = (v_x, t_x, \mathrm{last}(x))$ and the creation/completion rule of Section 4.2. A public read makes a real query to x in a table instance exactly when $\mathrm{last}(x)$ precedes its creation, probes all active tables even after a hit, compares the returned records, buffer, and cache, and stores the largest version in $C[x]$. Any records consumed during this operation are therefore dominated by the new cached version, and cached versions do not decrease. This proves the invariant

a record removed by a completed public lookup of x has version at most that in $C[x]$.

A record removed during a merge is instead dominated by the newer record retained by that merge; if that newer record is later consumed publicly, the cache then dominates both. Consequently, while the index is clean, every current logical value is represented by a live index record or by the public cache. Skipped tables cannot contain an uncached newer value, since their key was already queried after their creation. This proves repeated-public-read correctness and nonrecurrence, including misses. Cache records never enter carry inputs; index discards and full resets are treated in Section B.4.

Lemma B.5 (Preservation of the joint shuffle invariant)**.** *In the ideal-primitive execution, Procedures 1–4, initialization, and resets satisfy* (23) *at every stated interface, including under feedback from prior traces and logical responses.*

Proof. We check each way in which a shuffled-input interface can be reached. Initialization shuffles the canonical input with fresh coins. A buffer flush first determines its records from the logical history, then shuffles them locally. The two hierarchies use independent permutations even when their records coincide; no shuffled copy is shared.

Lookups and buffer deduplication select logical records by keys and versions independent of their physical order, and responses depend on logical values. The next application request is a function of $\mathcal{H}$ and its own coins, so it cannot reveal a still-hidden permutation through a representation-dependent choice. A previously queried table retains the state of its reactive instance; however, its retiring extraction supplies a fresh residual permutation.

For a merge, Lemma B.4 proves (23) at the temporary build, at the final interspersing, and at the completed merge interface; its complete scan makes the residual multiset independent of query order. The normalization of Section B.1 ensures uniform order of real records and hides the dummy positions, so building the destination table also satisfies (23). Each source table belongs to one hierarchy and is retired before its residual is used.

Discards depend only on public occupancy counters. Canonical export has a public trace, retains only the logical canonical array, and leaves live tables unchanged (Lemma B.6), so it reveals no layout information to the execution continuation. Any new authority is independently shuffled before its first build, just like at initialization, and the index begins with an empty hierarchy after a reset. These cases close the induction at the interfaces in (23). □

Privacy. Every table instance is used admissibly in the sense of Lemma B.2: the two correctness arguments above give nonrecurrence, and Lemma B.5 gives the shuffle condition (23). The rest of the trace, outside the table instances' own accesses, is determined by public information: schedules and padded lengths depend only on modes, counters, and, for W, disclosed kinds; plain accesses use public addresses. Theorem B.9 combines these facts into the full adaptive simulation.

Peak storage accounting. Let S_{PRF} be the total number of bits occupied by the PRF's shared key and peak evaluation workspace. Each tagged record occupies a fixed number of words. Table 4 accounts for the total storage required at any given time; the two hierarchies and all maintenance procedures run sequentially, so one temporary workspace suffices.

There are $O(\log_2 n)$ live tables and $O(1)$ additional temporary tables, each contributing two $O(\log_2 \kappa)$-record final stashes. Local deduplication and shuffling reuse $O(q)$ buffer workspace. The imported primitives use the storage placement of Lemma B.2, and export uses constant-size payload registers and public control information. Thus the peak client state in bits satisfies, for fixed constants independent of the growth factor and horizon,

$$M_{\mathrm{client}} \le C_{\mathrm{buf}}(q+\Delta)w + C_{\mathrm{stash}} \log_2(2+n) \log_2(2+\kappa)w + C_{\mathrm{ctl}} \log_2(2+n)w + S_{\mathrm{PRF}}. \tag{25}$$

Since $\kappa \le n$, $q \ge \log_2^{12}(2+n)$, and $\Delta = O(\log_2^2 n)$, the noncryptographic terms total $O(qw)$, and the assumed $S_{\mathrm{PRF}} = O(qw)$ gives $O(q)$ client words overall. For the exact mw-bit cap, reserve the PRF fraction first and choose q as a sufficiently small constant fraction of the remaining word budget, then choose C_0 to preserve the primitive threshold (Appendix G). Geometric capacities and sequential scratch reuse give $O(n)$ server words, including the new destination during a carry.

Component	Location	Peak words
Authority and index buffers, local deduplication, and batch shuffling	Client	$O(q + \Delta)$
Final stashes of live and temporary consuming tables	Client	$O(\log_2 n \log_2 \kappa)$
CPU registers, level and instance counters, and caller state	Client	$O(\log_2 n)$
Shared PRF key and evaluation workspace	Client	$\lceil S_{\mathrm{PRF}}/w \rceil$
Live tables, public cache, and construction metadata	Server	$O(n)$
Merge arrays, new destination instance, and export/sort/shuffle scratch	Server	$O(n)$

Table 4: Storage allocation, including transient state. Server scratch is accessed through probes that are accounted for. Only the final stashes of each table are retained locally; the combined small-bin stash table stays on the server.

B.4 Canonical export, resets, and cost allocation

Lemma B.6 (Nonmutating export with a small client)**.** *Either hierarchy (the index only if clean) can export its current logical array in canonical address order using $O(n \log_2 n)$ probes, $O(n)$ server scratch, and the same $O(q)$ client bound. Export is private jointly with future lookups and leaves the live tables unchanged. A fresh authority or index can be initialized from the array at the same cost.*

Proof. Scan the payload arrays of all live tables in their fixed physical order, and copy their retained stashes and the client buffer into padded scratch. The consuming-table representation stores payloads explicitly in its main arrays and its two retained stashes; ignore build metadata and retired scratch. Reading all these public-size arrays collects every live record without invoking extraction or revealing which payload slots are real, and it is harmless to include an old payload copy, since versions select the current record. For the index include the public cache. Add one default version-zero record for each logical address; genuine versions start above zero, and a missing authority value after a failure is interpreted as 0^w before the next append.

Use a fixed oblivious sorting network on these $O(n)$ records, sorting by address and decreasing version, then keep the first record of each address in a fixed scan, and obliviously compact and sort the retained records into canonical order. The $O(n \log_2 n)$-size sorting networks of Ajtai, Komlós, and Szemerédi [AKS83] suffice (see also [AKL+20, Theorem 4.1]); their comparators use a constant number of payload registers, with public control stored within the workspace allowance. The correctness invariants above ensure that every current version is collected, and the default records ensure a defined output on failure paths, where malformed logical addresses are marked dummy. Use scratch regions disjoint from every live table. After producing the canonical output, overwrite the other scratch in fixed full scans and clear temporary local records; no raw payload order, slot mark, or layout-dependent intermediate information is retained by the caller.

Why the live table simulators can continue. The export reads outside the table API, so its security is justified directly rather than inferred from the lookup/extraction theorem. On the good execution, export has three properties: its directed trace is fixed by the public capacities; its only retained output is the canonical logical array; and it leaves every live table's cells, stashes, and private control state unchanged. The sorting network and fixed scans establish the first, version selection the second, and disjoint scratch the third. In particular, the export retains no information from the scan that would allow subsequent lookups to depend on the physical table layout. At the export boundary, couple the live table state before and after the scan identically, append the

public export trace to the simulated transcript, return the canonical array through the ideal logical computation, and retain each existing table simulator's state unchanged. The next table call is then a continuation of the same reactive instance. The simulator need not reconstruct scanned payload bytes, because cell contents are absent from the trace and no layout-dependent output escapes the export. This proves joint simulation with later calls, including calls chosen from subsequent application responses; the argument concerns the whole experiment up to the first primitive failure, and bounded continuation contributes only its negligible error.

To initialize a fresh authority, tag every canonical value with the current global version, obliviously shuffle the array in $O(n \log_2 n)$ probes using [AKL+20, Theorem 4.6], and build its bottom table. To reset the index with exported authority contents, write them to the public cache at that same current version, clear the index's tables and buffer, and reset completion times to sentinel values. Do not reset global version, instance, or cryptographic domain counters. If resetting both authority and index, only the authority is built from a shuffled array; the index starts empty. Thus no shuffled input is reused by two live table copies. □

Proof of Theorem B.1. Use two independently represented hierarchies. Every private request queries the authority and appends the returned value on a read or the supplied value on a write; it appends the same new value and version to the index unless the index is dirty. A public write invokes Lemma B.6, changes its disclosed address to the new value and version, and resets both interfaces; its $O(n \log_2 n)$ probes are charged to that public write. Summing carry accounting over reset epochs preserves the private prefix bounds.

For schedule G (deferred private index work), the authority, with growth factor two, performs its carries immediately on private appends; its lookup and carry costs give $O(IH_q)$ private probes on every prefix with I private requests. The index accepts appends locally and performs its regular carries during public operations. Its dirty rule is as follows: when its buffer reaches $q + \Delta$, discard the index and suspend its appends; at the next public read, export the authority and reconstruct the public cache. This recovery costs $O(n \log_2 n)$ and does not retire or otherwise alter the authority. Correctness follows from the authoritative export, and no index lookup is performed while the index is dirty. Immediately after a clean public operation, index occupancy is below q, so a dirty transition requires at least Δ consecutive private modes, and at most Np^{Δ} transitions occur in expectation in a length-N prefix; recoveries are no more numerous. For $r = 1 - p \geq 1/2$ this gives

$$\mathbb{E}[\text{public-read probes}] = rNR_N \leq C(rNL + pNBL + n \log_2(2 + n)Np^{\Delta}),$$

and the recovery term divided by rN is negligible. Local index appends and dirty transitions make no private probes, proving the stated bound for schedule G.

For schedule I (privately paid index work), use growth factor two for the authority and B for the index, both executing carries immediately on private appends. Equation (24) gives at most $CI(H_q + BL)$ private probes on every prefix, and each public read costs $O(L)$. There is no dirty recovery, and the stated bounds for this schedule hold also for $1/2 < p < 1$.

For schedule A (all regular private carries deferred), both hierarchies use growth factor B, and the index follows schedule G. Public operations service a due authority carry before servicing the index. A full authority buffer forces a private carry, costing at most Cn probes and requiring Δ consecutive private modes. Hence $\Pi_N \leq C(L + np^{\Delta-1}) = O(L) + \mathrm{negl}(\kappa)$. Total regular carry work in both hierarchies is $O(FBL)$ after F private appends; adding the index's recovery term proves the stated bounds for this schedule.

Skipping the authority lookup on disclosed private writes removes the $O(L)$ private term on the all-write workload, giving schedule W; its nonrecurrence and correctness were proved above, and comparable disclosed-kind inputs skip authority lookups on exactly the same requests.

Resource bounds on every tape follow from Convention B.3, using default values on missing authority records and truncating malformed intermediate states to their allocated storage. For correctness and privacy, couple to the ideal execution until the first primitive error; in that execution all nonrecurrence, distinct-key, and shuffle preconditions hold, so a polynomial union bound and the shared-PRF hybrid give negligible total error. Theorem B.9 gives the interactive composition and shows that the same cost accounting holds under adaptive request selection. □

Corollary B.7 (Linear initialization and resets with polynomial workspace)**.** *If additionally $q \geq n^{\theta}$ for fixed $\theta > 0$, the same construction has $O_\theta(n)$ initialization and public-write cost, with all the guarantees of Theorem B.1.*

Proof. The external-memory oblivious sorter of Goodrich and Mitzenmacher [GM11, Theorem 6] sorts s records using $O((s/B_e)\log^2_{M_e/B_e}(s/B_e))$ I/Os under $M_e > 3B_e^4$. Set $B_e = 1$ and reserve $M_e = \Theta(q)$ client records. Arrays fitting in the client are processed locally; for $s = O(n)$ outside it, $q \geq n^{\theta}$ makes the logarithmic factor constant. Each tagged record occupies $O(1)$ cells. Thus the sorting and filtering in Lemma B.6 cost $O_\theta(n)$ probes with $O(q)$ client and $O(n)$ server space.

For initialization and rebuilding the authority from a canonical array, use the sorting-based oblivious random permutation of [AKL+20, Theorem 4.3]. Its cost is $O(T_{\mathrm{sort}}(n) + n) = O_\theta(n)$ and its failure probability at most $\exp(-\sqrt{n})$, negligible in κ. It assigns random labels, sorts, and handles the rare colliding subset with a bounded oblivious shuffle, and does not assume the canonical input is already shuffled. The extra labels occupy $O(1)$ words per record, and the existing bounded-failure convention applies. The resulting array meets the consuming-table build precondition; routine carries and mode schedules are unchanged. □

Initialization and the read frontier. Applying the growth-factor tuning of Appendix C with $q = \Theta(m)$ and $H_q = \Theta(H)$ gives Theorem 3.2, the disclosed-write variant, and the variants with small average-input budgets. Initialization contributes $O(n \log_2 n/N)$ to total average work; on $N \geq 4n/p$ and $m \leq n^{1-\varepsilon}$ this is $O_\varepsilon(pH) = O_\varepsilon(F_p)$, preserving the throughput theorem. If initialization is instead charged entirely to private requests, the explicit additive term is $O(n \log_2 n/(pN))$; it is not part of the stated Π_N or of the constructions' stronger pointwise P-budget guarantees.

B.5 Adaptive requests with unchanged costs

The construction admits an online simulator; no change to its data structures or schedules is needed. We prove this directly from the reactive security of the imported primitives.

Interactive security and its leakage. Let $\mathsf{F}_{\mathrm{RAM}}$ be the ideal full read/write functionality, initialized with an array X. It executes each logical request exactly and returns the answer, or a write acknowledgment, to its caller. For a request u_t, let ℓ_t be its mode and, if public, its complete request, including a public write's value; for disclosed-kind security, ℓ_t additionally contains the private read/write kind. These are the incremental versions of the two leakage functions in Section 2.2.

Definition B.8 (Adaptive trace simulation)**.** A nonuniform probabilistic polynomial-time environment $\mathcal{E}$ chooses X and auxiliary information independently of the implementation's coins. It observes the initialization trace before choosing its first request. After each completed request it receives the logical response and the physical trace so far, and may use both to choose the next request or to stop. All executions have prescribed polynomial bounds. In the real experiment the implementation supplies the responses and traces. In the ideal experiment, $\mathsf{F}_{\mathrm{RAM}}$ supplies the responses and a stateful online probabilistic polynomial-time simulator Sim supplies each trace

segment, given only the public parameters, initialization size, and successive leakages $\ell_1, \ldots, \ell_t$. Adaptive trace simulation of a construction $\mathcal{A}$ means that one such simulator satisfies, for every $\mathcal{E}$,

$$\mathsf{View}^{\mathrm{real}}_{\mathcal{E},\mathcal{A}}(1^\kappa) \;\approx_c\; \mathsf{View}^{\mathrm{ideal}}_{\mathcal{E},\mathsf{F}_{\mathrm{RAM}},\mathsf{Sim}}(1^\kappa),$$

where the view includes the environment's inputs, auxiliary information, coins, responses, and observed traces, including the environment's stopping decision.

The environment models a request-generating application together with the trace observer. Including responses in its view asserts security under feedback from private computation. Any information encoded by the application into a later public request is part of the declared leakage.

This definition also implies security in the following adaptive left-or-right trace-indistinguishability game. Both branches start from the same array and public parameters. A controller chooses, from the common observed history, a pair of requests with equal incremental leakage; a fixed hidden bit selects the executed branch. The controller sees its trace but not the response of the selected branch; however, it may maintain both ideal logical arrays and compute both branches' answers itself. Replacing either branch by the online simulator gives the same experiment because every submitted pair has equal leakage. Thus, the bit remains hidden even when the two branches' read answers differ. In particular, the fixed-sequence comparisons used by all our lower bounds are covered by this game.

Theorem B.9 (Adaptive extension)**.** *Under the parameter and PRF assumptions of this appendix, schedules* I, G, A *satisfy hidden-kind adaptive trace simulation, and* W *satisfies disclosed-kind adaptive trace simulation. Their whole-execution correctness error is negligible against every such environment. This holds for arbitrary adaptive modes, addresses, kinds, and values, including public writes. Space, initialization, public-write costs, and pointwise private prefix bounds are unchanged.*

At every fixed permitted horizon N, all the average-cost bounds of Theorem B.1 also hold uniformly over adaptive request choices, provided that each mode is a fresh independent Bernoulli draw with private probability $p \le 1/2$ and that public requests are reads. The mode sampler is independent of the full prior execution and implementation randomness; after observing the current mode, the environment may select the address, private kind, and value using all previous traces and responses. The schedule W *private-cost bound requires every private request to be a write. The polynomial-workspace improvement of Corollary B.7 also remains valid.*

Proof. Established interface conditions. Section B.3 proves logical correctness and nonrecurrence; Lemma B.5 proves (23), including the merge dependence handled in Lemma B.4. These results hold under adaptive feedback. We use the reactive contract of Lemma B.2 and the export continuation of Lemma B.6.

The simulator. Maintain the public mode counters, buffer occupancies, level digits, active-table identifiers, creation times, and dirty flags of both hierarchies; occupancy counts request slots even when keys repeat. For each table retain one primitive simulator throughout that instance's lifetime. From the next leakage item, determine all lookups, carries, discards, recoveries, and resets, together with their padded sizes and scratch locations. Private reads and writes have the same schedule in I, G, A; in W their disclosed kind determines whether the authority search is skipped. Real/dummy choices and lookup hits are hidden inputs to the primitive and need not be determined by the simulator. Generate the fixed scans and public-cache accesses directly, since their addresses depend on public sizes or the disclosed query address; invoke the relevant stateful simulators for table operations and the oblivious array primitives. Merge steps M1–M5 have public capacities and a fixed sequence of calls, and initialization is handled in the same way. All table identifiers are fresh, including after resets.

For public writes and dirty recovery, use the separate export argument of Lemma B.6: insert its public scan and sorting trace, keep each surviving table simulator in exactly its previous state, and pass only the canonical logical array to the reset procedure. Discard layout-dependent scratch. If a table is replaced, create a fresh simulator for its new identifier after an independent shuffle; if it survives, resume its existing simulator. This distinction is needed for dirty-index recovery, which preserves the authority. The faster export of Corollary B.7 has the same functional output and unchanged live state; its oblivious sorter supplies the corresponding jointly simulated scratch trace.

Composition and correctness. Replace the shared PRF by one lazily sampled random function; fresh domains separate dynamically created table and subroutine instances. Apply the contract of Lemma B.2 to complete table lifetimes, retaining simulator state across calls, and Lemma B.6 at exports; Lemma B.5 supplies every shuffled-input precondition. Each ideal replacement preserves the joint view of the application, the other instances, and the code selecting later calls, so it preserves their adaptive choices. The resulting trace is generated by the online simulator above from successive leakages alone, jointly with the ideal logical responses. A polynomial composition bound, the PRF advantage, and Convention B.3 give negligible error for the whole interactive experiment. Logical correctness follows from the hierarchy and cache invariants up to the first exceptional event, and the space and private prefix bounds hold on every tape, hence on every adaptive execution.

Cost under fresh IID modes. Let M_t indicate that request t is private, put $I_N = \sum_{t=1}^{N} M_t$, and define

$$Z_N = \sum_{t=\Delta}^{N} \prod_{j=0}^{\Delta-1} M_{t-j}, \qquad \mathbb{E}I_N = pN, \qquad \mathbb{E}Z_N \le Np^{\Delta}.$$

Z_N is the number of windows of Δ consecutive private requests. Every forced private carry and every dirty transition requires one of these windows, and recoveries are no more numerous than dirty transitions. On every execution, regular carry work is $O(I_N BL)$, public lookup work is $O((N - I_N)L)$, and recovery work is $O(n \log_2(2+n) Z_N)$. For schedule A, private work is at most $CI_N L + CnZ_N$; for schedule W on private writes the first term vanishes. Consequently

$$\Pi_N^{\mathrm{A}} \le CL + Cnp^{\Delta-1}, \qquad \Pi_N^{\mathrm{W}} \le Cnp^{\Delta-1},$$

and, for G, A, W,

$$R_N \le C\left(L + \frac{p}{1-p}BL + \frac{n\log_2(2+n)}{1-p}p^{\Delta}\right).$$

Schedule I has $R_N = O(L)$ and its original pointwise private bound; G retains the pointwise private bound of its authority with growth factor two. The displayed window terms are negligible for $p \le 1/2$. These estimates use only the fresh modes and deterministic accounting, so feedback through requests cannot change them. They hold at fixed N; the ratios Π_N, R_N are not assertions about conditioning on an adaptively chosen stopping time, although security and pointwise prefix bounds do permit such stopping. Faster export and initialization with polynomial workspace preserve the same simulation and accounting. □

Corollary B.10 (The frontier against adaptive request selection)**.** *With fresh IID modes and no public writes, require the private average-input budget Q uniformly over all adaptive request controllers of Definition B.8, and measure the worst-case public-read average over those controllers at a deterministic permitted horizon. Under the assumptions of Theorems 3.1 and 3.2, when measured this way, the optimal public-read cost among adaptively secure hidden-kind families is $\Theta(G_p(Q))$ for $Q \ge C_1 F_p$. With a worst-case-input private budget Q it is $\Theta(G_p(Q))$ for $Q \ge C_2 H$. The optimal worst-case total average is $\Theta(F_p)$, including initialization on the stated horizons.*

Proof. Theorem B.9 makes the existing growth-factor tuning uniform over these controllers, giving the upper bounds. A controller generating the IID uniform-address write workload is among them, and adaptive simulation implies its required fixed-sequence privacy, so the original lower bounds apply to that controller. All family quantifiers remain those of Appendix G. □

C Budget inversion and precise tuning

This appendix gives the budget tuning for the read frontier, proves the overall throughput and worst-case-input budget statements, and then states and proves the smaller-budget consequences. All constants may depend on the fixed memory exponents and the construction constants, but not on p, Q, or N.

Lemma C.1 (Uniform logarithmic inversion)**.** *For $A \geq 2$, $D \geq 0$, and $S > 0$, if $x \log_2(A + D/x) \geq cS$, then $x \geq c'S/\log_2(A + D/S)$ for a constant $c' > 0$ depending only on c.*

Proof. The function $f(x) = x \log_2(A + D/x)$ is increasing for $x > 0$: its derivative times $\ln 2$ is $\ln(A + z) - z/(A + z) > 0$, where $z = D/x$. Put $\lambda = \log_2(A + D/S) \geq 1$. For $0 < \delta \leq 1$,

$$f(\delta S/\lambda) \leq \frac{\delta S}{\lambda}(\lambda + \log_2 \lambda + \log_2(1/\delta)) \leq \delta S(2 + \log_2(1/\delta)).$$

Choose δ sufficiently small that the last expression is below cS and use monotonicity. □

Lemma C.2 (Choosing an integer growth factor)**.** *Let $s = n/q \geq 2$, $h = \log_2 s$, and $L = \lceil \log_B s \rceil$. For any fixed $c_* > 0$ there are constants C, D such that, for $V \geq Ch$, the integer*

$$B = \min\{\lceil s \rceil, \max\{2, \lfloor V/(Dh) \rfloor\}\}$$

satisfies

$$c_* BL \leq V, \qquad L = O\left(\max\left\{1, \frac{h}{\log_2(2 + V/h)}\right\}\right).$$

Proof. Take $D \geq 2c_*$ and $C \geq 4D^2$. Then $V/(Dh) \geq C/D \geq 2$, so the maximum with two is never active: $B = \min\{\lceil s \rceil, \lfloor V/(Dh) \rfloor\}$ and $2 \leq B \leq V/(Dh)$. Since $h \geq 1$, this gives $L \leq h/\log_2 B + 1 \leq 2h$ and hence

$$c_* BL \leq c_* \cdot \frac{V}{Dh} \cdot 2h = \frac{2c_*}{D} V \leq V.$$

For the second property, if $B = \lceil s \rceil$ then $L = 1$. Otherwise $B = \lfloor V/(Dh) \rfloor \geq V/(2Dh) \geq \sqrt{V/h}$, using $V/h \geq C \geq 4D^2$. Thus $\log_2 B \geq \frac{1}{2}\log_2(V/h) = \Omega(\log_2(2 + V/h))$, and $L \leq h/\log_2 B + 1$ gives the claim. □

Proof of Theorem 3.1. Appendix A supplies, on either workload, $R_N \geq c$ and

$$R_N \log_2\left(2 + \frac{Q}{R_N} + \frac{r}{p}\right) \geq cH \quad \text{if } \Pi_N \leq Q.$$

Apply Lemma C.1 with $A = 2 + r/p$, $D = Q$, and $S = H$. Next, replace the $2 + r/p$ term with $2 + 1/p$, which changes the logarithm by at most a constant factor. This proves $R_N = \Omega(G_p(Q))$. The same applies to a worst-case-input private budget, which implies the average-input budget on both workloads. □

Proof of Theorem 3.2. Use Theorem B.1 with $q = \Theta(m)$ and $H_q = \Theta(H)$, fixing all construction constants, including those in the memory allocation. Choose the growth factor

$$B_p = \min\{\lceil n/q \rceil, \max\{2, \lfloor 1/p \rfloor\}\}.$$

Then $pB_p = O(1)$ and the resulting depth satisfies $L = O(F_p)$—before clipping B_p this follows from $\log_2 B_p = \Theta(\log_2(2 + 1/p))$, and after clipping from $L = 1$. Schedule A consequently has $\Pi_N, R_N = O(F_p)$, and schedule G has $P = O(H)$ and $R_N = O(F_p)$.

For sufficiently large Q/H, taking schedule I and using Lemma C.2 we have $P \le Q$ and $R_N = O(\max\{1, H/\log_2(2 + Q/H)\})$. To account for the authority's $O(H)$ cost, allocate at most $Q/2$ to the index and choose the threshold constant in $Q \ge C_2 H$ large enough to fund the authority with the other half. Taking the better of schedules G and I yields $O(G_p(Q))$, since $\max\{\log_2(2 + 1/p), \log_2(2 + Q/H)\} = \Theta(\log_2(2 + 1/p + Q/H))$. Choose C_2 also large enough that Lemma C.2 applies whenever $Q \ge C_2 H$. This proves the worst-case-input budget upper bound and the upper bound for average-input budgets in this range. For $C_1 F_p \le Q < C_2 H$, $G_p(Q) = \Theta(F_p)$, so schedule A suffices; choose C_1 larger than its private-cost constant with a fixed margin, which absorbs the negligible additive cost. All the above choices of the growth factor are integers in the allowed range. □

Theorem C.3 (Total work and private-budget separation)**.** *Under the lower-bound assumptions of Theorem 3.1, every secure family has $T_N = \Omega(F_p)$, and every secure full-API implementation with a worst-case-input private budget P has $P = \Omega(H)$, even when its average-input private cost is much smaller. Under the assumptions of Theorem 3.2, there exists a hidden-kind implementation that has $\Pi_N, R_N = O(F_p)$, and another that has $T_N, R_N = O(F_p)$ while satisfying worst-case-input private budget $O(H)$. The matching overall throughput bounds also include the stated initialization cost on the permitted horizons.*

Average-input and worst-case-input budgets differ: at $p = n^{-c}$, both average costs can be constant, while every worst-case-input private budget is logarithmic.

Proof of Theorem C.3. Use the public-terminal lower bound in its form $R_N \log_2(2 + T_N/(pR_N)) \ge cH$. Monotonicity in R_N and $R_N \le T_N/r$ imply $(T_N/r)\log_2(2 + r/p) \ge cH$. Since $r \ge 1/2$, this gives $T_N = \Omega(H/\log_2(2 + 1/p))$; also $T_N \ge rR_N = \Omega(1)$, proving $\Omega(F_p)$. Schedule A at B_p has both average costs $O(F_p)$. Schedule G has $T_N \le O(pH + rF_p) = O(F_p)$, since $p\log_2(2 + 1/p)$ is bounded for $0 < p \le 1/2$, and its worst-case-input budget is $O(H)$. The small-client initialization adds $O(n \log_2 n/N)$ to total work, which is $O_\varepsilon(pH) = O_\varepsilon(F_p)$ for $N \ge 4n/p$ and $m \le n^{1-\varepsilon}$, so initialization does not change the optimal total-work bound; with polynomial client workspace, Corollary B.7 improves this contribution to $O(n/N) = O(p)$.

The worst-case-input budget claim follows from the total-work lower bound just proved. Given an implementation $\mathcal{A}$ with worst-case-input private budget P, construct a wrapper $\mathcal{B}$ that marks every incoming request private before passing it to $\mathcal{A}$, retaining $\mathcal{A}$'s parameters, including any tuning to p, and the original operation boundaries in the trace. On every fixed length-N input, $\mathcal{B}$ uses at most PN expected probes, so its total average is at most P. The wrapper preserves correctness and disclosed-kind privacy: two comparable external inputs have identical kinds at every position, so their all-private translations are comparable for $\mathcal{A}$; hidden-kind privacy of $\mathcal{A}$ also implies this disclosed-kind guarantee. Apply the total-work lower bound to $\mathcal{B}$ on $\mathcal{W}_{R,W}$ at $p = 1/2$ to obtain $P \ge T_N^{\mathcal{B}} = \Omega(H)$. In the computational case, the assumption that P is polynomially bounded and the stipulated preprocessing bound give the wrapper the required reference expected-probe bound. This also recovers the usual all-private ORAM obstruction [LN18] without a separate transfer of its security convention. □

Proof of Corollary 3.3. For fixed constant $0 < p \le 1/2$, the average-input budget $\Pi_N = O(H)$ allows us to take $Q = CH$ for a constant $C > 0$. Then $G_p(Q) = \Theta(H)$, so Theorem 3.1 gives $R_N = \Omega(H)$, including on $\mathcal{W}_{R,W}$ with disclosed private kinds. A constant bound on the average public-read probes on every fixed prefix without public writes would, after averaging over the workload, imply $R_N = O(1)$. This contradicts the lower bound since $H = \Theta(\log_2 n)$ under the stated memory assumption. □

C.1 Smaller private average-input budgets

Throughout this subsection fix $C_0 \log_2^{12}(2+n) \le m \le n^{1-\varepsilon}$ with $\varepsilon > 0$. Lower bounds use the assumptions of Theorem 3.1; constructions use those of Theorem 3.2. The next two results examine smaller private average-input budgets, not covered by the frontier, and how their feasibility depends on whether private operation kinds are disclosed.

Theorem C.4 (Small average-input budgets with hidden kinds)**.** *On either $\mathcal{W}_{R,R}$ or $\mathcal{W}_{R,W}$ with hidden private kinds, there are positive constants $Q_0, c_0, c_1, c_2, C_1, C_2$ such that for $Q_0 \le Q \le c_0 F_p$, every secure family with $\Pi_N \le Q$ satisfies $R_N \ge c_1 Q\, 2^{c_2 H/Q}$, and there exists a computationally secure family that attains $\Pi_N \le Q$ and $R_N \le C_1 Q\, 2^{C_2 H/Q}$. Note that for some parameter choices, the budget interval may be empty.*

Theorem C.5 (Small average-input budgets with disclosed writes)**.** *In the same resource regime, there exists a disclosed-kind computationally secure full-API family that has, on $\mathcal{W}_{R,W}$, $\Pi_N = \mathrm{negl}(\kappa)$ and $R_N = O(F_p)$. More generally, for every fixed $a > 0$ and every polynomially bounded $Q \ge \kappa^{-a}$, there is such a family with $\Pi_N \le Q$ and $R_N = O(G_p(Q))$. Together with Theorem 3.1, this gives a matching disclosed-kind public-read frontier throughout that range. Initialization remains separately charged.*

For constant p and a fixed constant $Q \ge Q_0$, hidden kinds therefore force $R_N = (n/(m+1))^{\Omega(1)}$ on both workloads, while disclosed kinds permit $R_N = O(H)$ on $\mathcal{W}_{R,W}$ with negligible average-input private cost. A worst-case-input private budget, which must also cover all-private inputs, is always $\Omega(H)$.

Proof of Theorem C.4. On either hidden-kind workload, Appendix A gives $Q \log_2(3 + rR_N/(pQ)) \ge cH$ if $\Pi_N \le Q$. Choose $Q_0 > c_0$. If $Q_0 \le Q \le c_0 F_p$, then $F_p > 1$ and $\log_2(1/p) \le c_0 H/Q$. Thus

$$R_N \ge \frac{pQ}{r}\left(2^{cH/Q} - 3\right) \ge c' Q\, 2^{c'' H/Q}$$

for sufficiently small fixed c_0, after absorbing the factor p and the subtractive constant. This proves the lower bound uniformly throughout the stated range.

For the upper bound fix a constant C_A so that schedule A satisfies $\Pi_N \le C_A L + \mathrm{negl}(\kappa)$ and $R_N \le C_A(L + pBL) + \mathrm{negl}(\kappa)$. Increase Q_0 so that $k = \lfloor Q/(2C_A) \rfloor \ge 1$ and $k = \Theta(Q)$, and take $B = \lceil (n/q)^{1/k} \rceil$. This is an allowed growth factor with $L \le k$ and $B \le 2 \cdot 2^{CH/Q}$. Hence the private cost is below Q for all sufficiently large parameters and $R_N \le C' Q(1 + p 2^{CH/Q}) \le C'' Q\, 2^{C''' H/Q}$. □

Proof of Theorem C.5. Use schedule W from Theorem B.1 with the clipped growth factor B_p from the proof of Theorem 3.2. Since $pB_p = O(1)$ and $L = O(F_p)$, it gives $R_N = O(F_p)$ and $\Pi_N \le Cnp^{\Delta-1} = \mathrm{negl}(\kappa)$. For every fixed $a > 0$, this private cost is at most κ^{-a} for all sufficiently large κ, uniformly in $p \le 1/2$ and the prescribed polynomial parameter range.

If $\kappa^{-a} \le Q < C_1 F_p$, where C_1 is the threshold constant of Theorem 3.2, then $Q/H = O(1)$ and $G_p(Q) = \Theta(F_p)$, so the same schedule attains $\Pi_N \le Q$ and $R_N = O(G_p(Q))$ throughout this

interval. For $Q \geq C_1 F_p$, use the hidden-kind constructions of that theorem, which also satisfy disclosed-kind privacy. Theorem 3.1 holds for every positive budget and supplies the matching disclosed-kind $\Omega(G_p(Q))$ bound. □

D Localized public-access charges and the read frontier

In this appendix, we first put the localized charges A_d and A_w in a common accounting framework, and then prove the localized read frontier. For that frontier we retain the two-class workloads $\mathcal{W}_{R,W}, \mathcal{W}_{R,R}$, $0 < p \leq 1/2$, $r = 1 - p$, uniform addresses, and the main horizon $N \geq 4n/p$.

D.1 A common accounting framework

Definition D.1 (Localized public-access charge)**.** Fix a deterministic horizon N. Choose a closing request $\tau \geq N$ from the request sequence, with $\mathbb{E}\tau < \infty$, and let c_i be the probes made by request i, excluding initialization. Specify a set $\mathcal{E}_\star \subseteq [\tau]$ of designated public accesses and a set $\mathcal{S}_\star \subseteq [\tau]$ of selected requests. The choice of both sets depends on the requests up to and including τ and only them. The localized charge $A_\star$ conveys how much work the requests in $\mathcal{S}_\star$ do per request in $\mathcal{E}_\star$. It is defined as follows, for $0 < \mathbb{E}D_\star < \infty$:

$$D_\star = |\mathcal{E}_\star|, \qquad C_\star = \sum_{i \in \mathcal{S}_\star} c_i, \qquad A_\star = \frac{\mathbb{E}C_\star}{\mathbb{E}D_\star}. \tag{26}$$

Average costs also fit this convention: take $\tau = N$ and let $\mathcal{E}_\star = \mathcal{S}_\star$ be all public reads or all public writes in the prefix to obtain R_N or W_N, respectively, when that class has positive probability. Localized charges A_d and A_w use the longer prefix and the selections in Table 5. Since the localized charges are only an accounting convention not used by the algorithm, they are allowed to select operations retrospectively, based on future operations. Their detailed definitions are Definitions D.3 and E.2.

	Read charge A_d	Write charge A_w
Workload	$\mathcal{W}_{R,R}$ or $\mathcal{W}_{R,W}$	$\mathcal{W}_{W,R}$
Designated accesses $\mathcal{E}_\star$	Each public read whose address has not been accessed after the previous private request	Each public write whose address will not be written before the next private request
Selected requests $\mathcal{S}_\star$	Each designated read and its preceding public request, if any	Each designated write and its next private request
Closing request	First public request strictly after N	First private request strictly after N
Accounting rules	No read before the first private request is designated	A private query shared by several designated writes is selected once

Table 5: The localized charges A_d and A_w.

Why the selection rules differ. A read must return its value immediately; its preceding public request is a place where public work could have prepared that read. A write may instead buffer its update until a later query. The private request closing a public-write phase may need to retrieve any surviving value from that phase, and it needs to do it obliviously, which means that in particular it has to obliviously handle any remaining buffered writes. For this reason, it is included in A_w.

Two examples. After a private request to x, suppose the next public reads are y, y, z, with $y \neq z$ and no earlier public request. The first and third reads are designated; all three are selected because the middle read precedes the third in the public subsequence. Here $D_d = 2$. For public writes to x, y, x followed by a private read, the second and third writes are designated; the second and third writes and the closing private read are selected, and $D_w = 2$.

Lemma D.2 (Selected and omitted work)**.** *Let $\mathcal{F}_i$ describe the requests and execution until the completion of request i, and set $q_i = \Pr(i \in \mathcal{S}_\star \mid \mathcal{F}_i)$ for $i \leq N$. Write $\bar{d}_\star = \mathbb{E}D_\star$ and $C_{\text{tail}} = \mathbb{E}[\sum_{N<i\leq\tau} \mathbf{1}_{\{i\in\mathcal{S}_\star\}} c_i]$. Then*

$$\bar{d}_\star A_\star = \mathbb{E}\sum_{i=1}^{N} q_i c_i + C_{\text{tail}}, \tag{27}$$

$$NT_N = \mathbb{E}\sum_{i=1}^{N} q_i c_i + \mathbb{E}\sum_{i=1}^{N}(1-q_i)c_i. \tag{28}$$

In particular, if $q_i \geq \gamma > 0$ whenever request i belongs to class b, with class probability f_b and average cost B_N, then

$$\gamma f_b N B_N \leq \bar{d}_\star A_\star. \tag{29}$$

Proof. The cost c_i is determined by $\mathcal{F}_i$, so conditional expectation gives $\mathbb{E}[\mathbf{1}_{\{i\in\mathcal{S}_\star\}} c_i] = \mathbb{E}[q_i c_i]$. Sum over the length-$N$ prefix, add the selected tail ending at request τ, and partition the prefix into selected and omitted work; nonnegativity permits Tonelli's theorem without a bound on the closing-tail cost. Restricting the prefix sum to class b proves (29). □

Bridging the localized charges and regular average costs. For reads, put $r = 1 - p$. Any public request i in the length-N prefix is selected with probability at least p, conditional on $\mathcal{F}_i$: with that probability a nonempty private run precedes its next public successor, so the successor is designated and selects the current request, and it occurs at or before the closing public request τ. Thus

$$prNR_N \leq \bar{d}_d A_d, \qquad \bar{d}_d = \Theta\big(rN\min\{1, pn\}\big) \quad (N \geq 4n/p).$$

Lemma D.4 gives the exact finite-prefix denominator. For writes, put $a = 1-p$, $\rho = p + a/n$, $\sigma = p/\rho$, and $c_N = 1 + 1/(pN)$. A completed public write is designated with probability σ, conditional on $\mathcal{F}_i$. If J is the expected total cost of selected private requests in the N-length prefix divided by apN, Lemma E.3 gives

$$\bar{d}_w = \sigma a(N + 1/p), \qquad \sigma a N W_N + apNJ \leq \bar{d}_w A_w, \qquad W_N + \rho J \leq c_N A_w,$$

and $B = p(\Pi_N - aJ)$ is the private work excluded from the charge per prefix request, which the identities in (36) retain explicitly.

For writes, Proposition E.17 gives a concrete implementation with tiny W_N and large A_w, thanks to private queries performing the work. Adding all omitted private probes gives the different charge A_w^+ of Definition E.2; it charges the helpers responsible for the gap between the optimal budget and cap.

D.2 Localized public-read charge and its accounting bounds

Definition D.3 (Localized public-read charge)**.** Instantiate Definition D.1 as follows. A public read is *bridging* if the read's address has not been accessed strictly between the most recent

private request and this read. Reads before the first private request are not bridging reads. Put $\tau = \min\{t > N : t \text{ is public}\}$, and let $\mathcal{E}_d$ be the bridging reads through τ. For each $t \in \mathcal{E}_d$, let $b(t)$ be the most recent public request strictly before t, or 0 if none exists, and put

$$\mathcal{S}_d = \left(\bigcup_{t \in \mathcal{E}_d} \{t, b(t)\} \right) \setminus \{0\}, \qquad D = |\mathcal{E}_d|, \qquad C = \sum_{i \in \mathcal{S}_d} c_i.$$

Define $A_d = \mathbb{E}C/\mathbb{E}D$, counting each selected request once. Both expectations include the entire closing tail.

Lemma D.4 (Counts and a predecessor-accounting bound)**.** *Put $z = r(1-1/n)$ and $d_0 = p/(1-z)$. For $n \geq 2$,*

$$\mathbb{E}D = rd_0 \left(N - \frac{z(1-z^N)}{1-z} \right) + p = \Theta(rN \min\{1, pn\}). \tag{30}$$

Moreover, at most $2D$ public requests are selected, and

$$\mathbb{E}C \geq prNR_N, \qquad \frac{T_N}{p} \leq \Pi_N + \frac{\mathbb{E}D}{p^2 N} A_d, \qquad \mathbb{E}D \leq rN + p \leq 2rN. \tag{31}$$

Proof. For a fixed terminal address at position j, the request at position $i < j$ is private with probability p, regardless of its address. Each request at a position strictly between i and j is public and avoids the terminal address with probability z. Summing over the position $i < j$ of the most recent private request, the request at position j is a bridging read with probability $rp \sum_{i=1}^{j-1} z^{j-i-1} = rd_0(1 - z^{j-1})$. Summing these probabilities over $j = 1, \ldots, N$ gives the expected number of bridging reads in the first N requests. If $\tau = N + 1$, its contribution is $rd_0(1 - z^N)$; otherwise the tail contains a private request, and the closing public read is always bridging, contributing p. This proves the exact $\mathbb{E}D$ formula; $pN \geq 4n$, $1 - z = p + r/n$, and $r \geq 1/2$, give the asymptotic result.

Every bridging read selects at most one predecessor. Conditional on the complete execution of a public request through N, its next public successor lies by τ; with probability p a nonempty private run intervenes, in which case the successor is a bridging read and the current request's probes are selected, regardless of whether it is itself a bridging read. Inequality (29) gives the first inequality in (31); substituting $T_N = p\Pi_N + rR_N$ gives the second, and the last follows from (30) and $N \geq 4n/p$. □

D.3 Lower bound for the localized public-read charge

Theorem D.5 (Localized public-read lower bound)**.** *Under Theorem 3.1's assumptions and its kind-disclosure conventions,*

$$A_d = \Omega(1), \qquad A_d \log_2 \left(2 + \frac{\Pi_N}{A_d} + \frac{r}{p} \right) = \Omega(H). \tag{32}$$

Consequently, $\Pi_N \leq Q$, or a worst-case-input budget $P \leq Q$, implies $A_d = \Omega(G_p(Q))$. Computational lower bounds require polynomial expected probes only on the original length-N reference workload, including initialization; they need no expected-cost assumption on the closing tail.

Proof. The constant floor, including rare private requests. Choose an endpoint e uniformly in the second half of the prefix. Preserve its last $f = \lfloor n/128 \rfloor$ requests and condition on their containing a private request, an event of probability $g = 1 - r^f = \Theta(\min\{1, pn\})$. Let S be the addresses accessed

privately before this window but not accessed anywhere in the window. For a fixed endpoint put $t = e - f \geq N/2 - f$, and let Z count the addresses not accessed privately in those t earlier positions. Each position privately accesses a fixed address with probability p/n, so

$$\mathbb{E}Z = n(1 - p/n)^t \leq n \exp(-pt/n) \leq n \exp(-2 + 1/256) < n/7,$$

using $N \geq 4n/p$, $p \leq 1/2$, and $f \leq n/128$ for the second inequality. Removing the window's addresses excludes at most f more addresses, so $|S| \geq n - Z - f$, and Markov's inequality gives $\Pr\{|S| < n/2\} \leq \mathbb{E}Z/(n/2 - f) \leq 128/441 < 1/3$. The same bound holds conditional on the window containing a private request, because Z depends only on the earlier requests; thus the joint conditioning used below has probability at least $2g/3$.

Replace all private requests before the window by fresh uniform writes at their original addresses. This preserves disclosed kinds on $\mathcal{W}_{R,W}$ and uses hidden kinds on $\mathcal{W}_{R,R}$. Conditional on modes and addresses, the final source values at S are independent uniform words: each is the word at its last private occurrence before the preserved window. Choose a uniform address of S after the prefix and issue the same public read in the altered and reference inputs; it is a bridging read in both executions. A message containing only the preterminal client state cannot recover a uniform coordinate of these $|S|$ words with constant success, by Lemma A.2, since $m/|S| = o(1)$. Simulation up to the first physical read (with the finite timeout of Convention A.4) and correctness therefore force a constant-probability physical-read event, which privacy transfers to the reference.

Let C^{own} count bridging reads' own probes through τ. The reference query has conditional address density at most $2/n$, and the conditioning has probability $\Omega(g)$, so its expected terminal cost is at most $O(\mathbb{E}C^{\mathrm{own}}/(grN))$. Hence $\mathbb{E}C \geq \mathbb{E}C^{\mathrm{own}} = \Omega(grN)$. Equation (30) has the same factor g, proving $A_d = \Omega(1)$.

The logarithmic bound. If pn is below a sufficiently large fixed constant, the floor and $\log_2(2 + r/p) = \Omega(\log_2 n) = \Omega(H)$ already suffice. Otherwise use the information window $\omega = \lfloor n/(128p) \rfloor$ and shortest epoch $\ell_0 = \lceil C(m+1)/p \rceil$ from Section 4.1. Also preserve the final private request in the last $f' = \lceil 4/p \rceil$ positions. Condition on this freshness window containing a private request and on a safe set of size at least $15n/16$ after excluding all reference-write addresses in the information window and all addresses in the freshness window. The expected excluded size is at most $n/128 + f'$; Markov's inequality and $r^{f'} \leq e^{-4}$ give constant probability after increasing the fixed threshold on pn.

Use the fixed-query coupling of Lemma 4.1, skipping the preserved last private request among source positions; increasing C absorbs this one omitted position. Insert distinct fresh words at safe addresses. Every unchanged write avoids these addresses, and the preserved last private request and subsequent public reads do too, so the source word survives and both terminal reads are bridging. The repair proof and Lemma A.6(ii) give, for the reference terminal cost Y,

$$Y \log_2(2 + T_N/(pY)) = \Omega(H), \qquad Y = O\left(\frac{\mathbb{E}C}{rN}\right) = O(A_d),$$

the last inequality by (31). Monotonicity and that same accounting bound imply $A_d \log_2(2 + \Pi_N/A_d + 2r/p^2) = \Omega(H)$. For $p \leq 1/2$ the logarithm here is at most a fixed constant times $\log_2(2 + \Pi_N/A_d + r/p)$, proving (32); Lemma C.1 yields the frontier lower bound.

All lower-bound experiments use only the first N requests. Including the closing tail can only increase the expected charged cost $\mathbb{E}C$, which is the numerator of A_d. In the floor argument g is inverse-polynomial because $N \geq 4n/p$ is polynomial. Forcing a public terminal and restricting its address increase the reference expected probe cost by only constant factors. Prefix conditioning contributes at most a polynomial factor. Hence the assumed polynomial expected-probe bound for the original length-N workload suffices for Convention A.4. □

D.4 Matching constructions and the closing public request

The read constructions use counters sized for prescribed polynomial executions, whereas the localized objective has an unbounded geometric closing tail. We give an explicit continuation so that this tail is counted with bounded state, without changing the measured private prefix or introducing an expensive private fallback.

Lemma D.6 (A bounded continuation for the localized read objective)**.** *For a prescribed N, each schedule* $\mathrm{I}, \mathrm{G}, \mathrm{A}$ *of Theorem B.1 has a full-API realization that agrees with it through N, uses $O(q)$ counted client and $O(n)$ server words, and has a polynomial per-request probe cap throughout the closing tail. It preserves the schedules' asymptotic worst-case-input private budgets when promised. A constant number of retained PRF keys and one shared evaluator are counted in the client allowance. The expected additional public work before or at τ is negligible, including after normalization by $\mathbb{E}D$.*

Proof. Let $K = N + \lceil \log_2^2(2+n) \rceil$. Execute the schedule unchanged for K requests, with its counters sized for this polynomial interval. Then discard the public index and suspend its appends. Until the next public request, serve private requests using only the authority and its existing maintenance rule. That public request exports the authority, applies its own request, and resets both structures at cost $O(n \log_2 n)$; subsequent execution starts a new length-K interval. This is a deterministic rule based on disclosed modes and counters. Discarding the index costs no probes and leaves the authority correct, and the reset uses Lemma B.6 and is paid publicly.

During a possibly unbounded private continuation, apply the authority renewal rule of Lemma E.24: a full bottom carry retires every older authority table; after selecting the newest record per address, retag its output with version one and renumber any retained buffer suffix above it. Use a fresh key for the rebuilt bottom and subsequent tables, with table identifiers unique within that key's era. At most $O(n)$ appends occur before the next full bottom carry. These retags add only linear work to an already linear carry, also with polylogarithmic workspace; no canonical sorting is needed. Before the first such renewal, counters are bounded by $\mathrm{poly}(K+n)$; afterwards they are bounded by $\mathrm{poly}(n)$. The index has been discarded, so it has no versions or keys to keep consistent with this retagging. At a public reset all old tables retire; retag the canonical snapshot with version one and rebuild under fresh keys. Set the global version counter to one, and restart the separate event clock and cache completion sentinels. Subsequent updates receive versions at least two; version zero remains reserved for the default records in Lemma B.6, so a later export always prefers a genuine snapshot record to its default. Only a constant number of keys coexist, and calls share one evaluator.

The authority's nonrecurrence and shuffle invariants are unchanged, and the same carry accounting preserves worst-case-input private budgets up to a constant factor, with this factor absorbed by tuning the original schedule to a sufficiently small constant fraction of the budget Q. All the K-length cycles have bounded counters and polynomial per-request costs, including bounded primitive-failure continuations. A multi-key hybrid argument over the polynomially many independent keys gives adaptive security on prescribed polynomial-length executions; whole-execution correctness through a polynomial cutoff and the following tail estimate suffice for the stopped experiment.

Since $r \geq 1/2$, $\Pr\{\tau > K\} = p^{K-N}$ is negligible. For a polynomial cap M_*, all work beyond K has expectation at most $M_* p^{K-N}/r$. The prefix through N is unchanged, and the only extra public reset before τ occurs on this negligible event. Equation (30) bounds $\mathbb{E}D$ below by a positive constant on the stated horizons, so normalization preserves negligibility. □

Theorem D.7 (Matching localized-read frontier)**.** *Under the counted computational resources of Theorem 3.2, reserving a constant number of keys as in Lemma D.6, the optimal A_d on either workload*

is $\Theta(G_p(Q))$ for average-input budgets $Q \geq C_1 F_p$ and for worst-case-input budgets $Q \geq C_2 H$. The same constructions attain $R_N = O(G_p(Q))$. They have the read construction's server-space, initialization, and client-memory bounds and adaptive security. The A_d cost claim uses the specified IID uniform-address workloads.

Proof. Theorem D.5 supplies the lower bounds. Use the schedules of Theorem B.1 with Lemma D.6. At most $2D$ public requests are selected, so their lookup work is $O(LD)$. This is all schedule I pays publicly, apart from the negligible continuation cost, giving $A_d = O(L)$.

For G, A, regular carry work through τ is at most $CBLF_\tau$, where F_τ is the private-request count; each completed carry is funded by these appends, and discarded index potential only reduces the work. For each i, the event $\{i \leq \tau\}$ is determined by the modes before request i, so it is independent of whether request i is private. Hence $\mathbb{E}F_\tau = p\mathbb{E}\tau = p(N + 1/r)$. Dirty recovery costs $O(n \log_2 n)$ and requires $\Delta = \lceil \log_2^2(2+n) \rceil$ preceding private modes. There are at most Np^Δ expected recoveries in the initial prefix, and at the closing read, summing over its tail length bounds the recovery probability by $O((\Delta+1)p^\Delta)$. Thus all recovery work has negligible expectation, also after normalization. Equation (30) therefore gives

$$A_d = O\left(L + \frac{pBL}{r \min\{1, pn\}}\right) + \mathrm{negl}(\kappa).$$

For $p \geq 1/n$, choose $B = \Theta(1/p)$ clipped at $\lceil n/q \rceil$: $pB = O(1)$ and $L = O(F_p)$, giving $A_d = O(F_p)$. For $p < 1/n$, choose $B = \lceil n/q \rceil$ and $L = 1$. Now $\min\{1, pn\} = pn$, so the maintenance contribution is $O(B/(rn)) = O(1/q)$. Hence $A_d = O(1)$, which is again $O(F_p)$ because $F_p = \Theta(1)$ in this range.

Thus schedules G and A attain $A_d = O(F_p)$ for every $0 < p \leq 1/2$. The continuation leaves the first N requests unchanged, so Theorem B.1 still gives $\Pi_N = O(F_p)$ for schedule A and $P = O(H)$ for schedule G. Both schedules also retain the bound $R_N = O(F_p)$.

For $Q \geq C_2 H$, the tuning in Appendix C gives schedule I with $P \leq Q$ and

$$A_d, R_N = O\left(\max\left\{1, \frac{H}{\log_2(2 + Q/H)}\right\}\right).$$

Choosing the smaller of this bound and the $O(F_p)$ bound from schedule G gives $A_d, R_N = O(G_p(Q))$. This proves both budget claims when $Q \geq C_2 H$. For the remaining average-input budgets $C_1 F_p \leq Q < C_2 H$, we have $G_p(Q) = \Theta(F_p)$, so schedule A satisfies $\Pi_N \leq Q$ and gives the same bounds on A_d and R_N, provided C_1 is sufficiently large. □

E Public writes: lower bounds, constructions, and memory

This appendix treats the workload $\mathcal{W}_{W,R}$: a request is a public write with probability $a = 1 - p$ and a private read with probability p, where $0 < p \leq 1/2$; addresses are independent uniform elements of $[n]$ and public-write values are independent uniform w-bit words. Sections E.1–E.3 prove the lower bounds, Sections E.4–E.9 give the constructions matching them, and Section E.10 treats memory. Table 6 gives the route from each write result of Section 3.2 to the lemmas it uses; every construction additionally relies on the bounded-state Lemmas E.24 and E.25 and on Convention B.3.

The lower bounds already apply when only the public-write/private-read interface is supported; the constructions support the full API. The two-class proofs use only *public* writes as their entropy source, so their privacy comparisons preserve private read/write kinds and permit disclosed kinds. The lower-bound memory, word-size, correctness, and computational qualifications are those of Theorem 3.1, applied to $\mathcal{W}_{W,R}$; the horizon is $N \geq 4n$, with $1/p$ polynomially bounded in the

Result	Lower bound via	Construction via
Public write lower bound (Theorem 3.4)	Lemma E.1 ← Lemma A.6 and the fixed-query machinery of Appendix A	—
Private-budget threshold $\Theta(F_p/p)$ (Corollary E.19)	Lemma E.1	Corollary E.18 ← Theorem E.14 (private service) ← Lemmas E.11, E.12, E.13
Localized write charge bounds (Theorem E.9)	Corollary E.7 ← Theorem E.4 ← Lemmas E.3, E.1	Corollary E.16 ← Theorem E.14
Memory for small A_w	Theorem E.26 (from correctness only)	Corollary E.28 ← Theorem E.27, Corollary E.16
Low budgets and caps (Theorem E.10(i))	Corollary E.6 ← Theorem E.4	Corollary E.15 ← Theorem E.14 (eager service)
High budgets (Theorem E.10(ii))	Corollary E.5 ← Theorem E.4	Corollary E.23 ← Theorem E.22 ← Lemmas E.20, E.21
Buffering example	—	Proposition E.17

Table 6: Route from the public-write results of Section 3.2 to the results they use. An entry $A \leftarrow B$ means that the proof of A uses B.

computational case. The costs Π_N and W_N are averaged over the first N requests, with N fixed in advance, and

$$T_N = aW_N + p\Pi_N, \qquad H = \log_2(2 + n/(m+1)).$$

We state algebraic tradeoffs for finite costs (an infinite charged cost makes the corresponding lower bound immediate). Recall from Section 3.3 the symmetric, homogeneous, coordinatewise increasing function

$$\Phi(x,y) = \min\{x,y\}\log_2\left(2 + \frac{\max\{x,y\}}{\min\{x,y\}}\right) \quad (x,y>0),$$

with value zero on the axes. Let J be the expected private probes at positions $2,\dots,N$ immediately preceded by a public write, divided by apN; the preceding write is part of the conditioning, and J need not equal Π_N.

E.1 An average-cost tradeoff with disclosed private kinds

Lemma E.1 (Extended public write lower bound)**.** *On the workload $\mathcal{W}_{W,R}$, with disclosed or hidden private kinds,*

$$T_N, \Pi_N, W_N + J = \Omega(1), \qquad \Phi(T_N, \Pi_N) = \Omega(H), \qquad \Phi(T_N, W_N + J) = \Omega(H). \tag{33}$$

The constants are independent of p. In particular, the first tradeoff concerns the average public-write cost W_N, without retrospective labels or omitted maintenance.

Proof of Lemma E.1. Choose e uniformly from $\{\lceil N/2\rceil,\dots,N-2\}$ and draw the reference prefix through e. Append either a single private read, or a public write followed by a private read, with independent reference addresses (and value for the write), and denote the expected cost of this terminal block by Y. Online execution and the constant-size endpoint range give, respectively,

$Y \le 4\Pi_N$ or $Y \le 4(W_N + J)$: for the first request of the pair, conditioning its successor to be private does not change its cost, and for the second, the normalization apN in J accounts for the forced pair.

Set the window $\omega = \lfloor n/1024 \rfloor$ and $\ell_0 = \lceil C(m+1)/a \rceil$. For any fixed epoch I of length $\ell \ge \ell_0$ in the window before e, let E_I count dependency edges from I into the younger prefix and D_I those from I into the terminal block, where an edge joins consecutive probes to one physical cell when the second is a read. We claim

$$\mathbb{E}D_I + \frac{\mathbb{E}E_I}{\eta a \ell} \ge c \tag{34}$$

for absolute $c, \eta > 0$ and sufficiently large C.

Let G be the event that I contains at least $a\ell/2$ public writes; its probability is at least $63/64$. Conditional on G, select the first $k = \lfloor a\ell/4 \rfloor$ public writes of I. Their values $X_1, \dots, X_k$ are independent uniform words; denote their addresses by $A_1, \dots, A_k$. Keep the entire prefix unchanged. After the prefix has been specified, choose independent uniform $J_0 \in [k]$ and let the altered terminal private read query A_{J_0}; the reference terminal instead queries an independent uniform address, and the earlier public write of a two-request terminal block is identical on both sides. Every public request, including each X_j, and every private kind agrees pointwise in this pair, so disclosed-kind privacy suffices. There are at most $\omega + 1$ subsequent positions that can overwrite a chosen source word, and their uniform addresses are independent of its address, so a union bound gives overwrite probability at most $(\omega + 1)/n < 1/256$ for large n; conditioning on G does not affect address independence. Hence the altered terminal answer is X_{J_0} with probability at least $2/3 - 1/256$.

The entropy decoder is given modes, addresses, values outside these k source writes, the pre-epoch state, and implementation tapes, but *not* the k source values—public disclosure to the server does not make those words free side information for this decoder. All this side information is independent of the source words and J_0. Before J_0 is revealed, send the post-epoch client state and $a_I = \lfloor \eta a \ell \rfloor$ padded cell-repair records, aborting when $E_I > a_I$; for small η and large C the message has at most $kw/16$ bits. The reconstruction in Appendix A.2 applies verbatim: repairs reproduce the younger prefix, and if $D_I = 0$ they also reproduce the entire terminal block, with the finite simulation cap of Convention A.4(d) losing at most $1/64$. Lemma A.2 therefore forces $\{E_I > a_I \vee D_I > 0\}$ with constant probability in the altered execution. This trace-observable event transfers to the paired reference by Lemma A.3. Convention A.4 applies: G is sampled by rejection, and forcing the terminal request classes increases the reference experiment's expected probe count by at most a factor $1/(ap)$, up to constant factors from conditioning. The original reference workload has polynomial expected probe count, so the transfer requires no correctness guarantee stronger than $2/3$. Remove the constant-probability conditioning on G and use Markov's inequality to prove (34).

We now apply Lemma A.6 with density $d = a$, the window ω and shortest epoch ℓ_0 above, $g = 1$, and the above terminal block; since $a \ge 1/2$, $N \ge 4n$, and $m + 1 \le C_{\mathrm{mem}} n^{1-\varepsilon}$, every endpoint is preceded by a full window and $\omega \ge 2\ell_0$ for large n, the endpoint range has at least $N/4$ positions, $h_* = \log_2(2 + \omega/\ell_0) = \Theta(H)$, and $X = T_N/a = \Theta(T_N)$. Parts (i) and (ii) give $Y = \Omega(1)$ and the tradeoff inequality. Part (iii) needs the floor $X = \Omega(1)$, which holds already by correctness: take disjoint length-ω intervals. Their final surviving public-write values contain $\Omega(\omega w)$ expected independent information. The complete state after an interval recovers each survivor by a separate appended query. The post-interval client state and the address/value list of changed server cells, with a unary-encoded list length, form a prefix-free message of expected length $mw + O(w\mathbb{E}[\text{physical writes}]) + O(1)$. Coordinatewise Fano bounds and the entropy lower bound on expected prefix-free code length [CT06, Chapters 2 and 5] require $\Omega(\omega w)$ bits. Since $\omega \gg m$, each interval makes $\Omega(\omega)$ expected physical writes; summing gives $T_N = \Omega(1)$. Lemma A.6(iii) now gives $\Phi(X, Y) = \Omega(H)$, and monotonicity with the two upper bounds on Y proves (33). □

Proof of Theorem 3.4. The bounds $T_N, \Pi_N = \Omega(1)$ and $\Phi(T_N, \Pi_N) = \Omega(H)$ of Lemma E.1 give (6), with constants independent of p. The lemma's proof uses only public writes and private reads and preserves all private kinds, so it also applies to the restricted interface and permits disclosed kinds. □

E.2 Localized write charges and exact helper accounting

A public write is *bridging* if it is the last public write to its address before the next private request. Note that this is a retrospective label unavailable to the implementation. A private request is *selected* when immediately preceded by a public write; it closes a public phase containing a write.

Definition E.2 (Localized public-write charge)**.** Instantiate Definition D.1 as follows. Fix the prefix length N and let $\tau > N$ be the first private request after position N. Let $\mathcal{E}_w$ be the bridging writes through τ. For each $t \in \mathcal{E}_w$, let $b(t)$ be the next private request and put

$$\mathcal{S}_w = \bigcup_{t \in \mathcal{E}_w} \{t, b(t)\}, \qquad D = |\mathcal{E}_w|, \qquad C = \sum_{i \in \mathcal{S}_w} c_i.$$

Let C^+ be C plus the probes of all remaining private requests. Define

$$A_w = \frac{\mathbb{E}C}{\mathbb{E}D}, \qquad A_w^+ = \frac{\mathbb{E}C^+}{\mathbb{E}D}, \qquad \rho = p + a/n, \quad \sigma = p/\rho, \quad c_N = 1 + 1/(pN).$$

Let Γ be the expected probes on private requests in the initial prefix that are either first or immediately preceded by a private request, divided by N. These are the private *helpers omitted by* A_w. For a uniformly chosen position in the prefix, write $\mathcal{P}$ for its being private, $G \subseteq \mathcal{P}$ for its being an omitted helper, and C_0 for its private cost (zero at public positions). Then

$$e_N := \Pr(G) = p^2 + ap/N, \qquad \Gamma = \mathbb{E}[C_0 \mathbf{1}_G], \qquad \mathbb{E}[C_0 \mid \mathcal{P}] = \Pi_N.$$

Lemma E.3 (Exact counts and allocation)**.** *If Z counts selected private requests through τ, then*

$$\begin{aligned} &\mathbb{E}\tau = N + 1/p, \qquad \mathbb{E}D = \sigma a(N + 1/p), \qquad \mathbb{E}Z = ap(N + 1/p), \\ &\mathbb{E}C = \sigma\mathbb{E}[\textit{all public-write probes through } \tau] + \mathbb{E}[\textit{selected private probes through } \tau]. \end{aligned} \tag{35}$$

In particular, putting $\bar{A} = c_N A_w$,

$$W_N + \rho J \le \bar{A}, \qquad \Gamma = p(\Pi_N - aJ), \qquad T_N \le a\bar{A} - a^2 J/n + \Gamma. \tag{36}$$

For every N, the contribution to A_w from selected private requests is at most $P\rho/a$ under a worst-case-input private budget P, and at most $K\rho$ under a pointwise private cap K or an expected cap at every private position of every fixed input.

Proof. After a completed public write to address x, there is a race between a private request (probability p per request) and another public write to x (probability a/n per request). The private request wins with probability $\sigma = p/(p + a/n)$, exactly when the write is bridging. This remains its conditional probability after conditioning on the write's execution and probe cost. Moreover, whether the write lies in the prefix stopped at τ is known when it completes. Conditioning on the history through the completion of each public write and summing its cost times its bridging indicator therefore gives the thinning identity

$$\mathbb{E}[\text{probes on bridging writes through } \tau] = \sigma\,\mathbb{E}[\text{probes on all public writes through } \tau].$$

Tonelli's theorem permits the sum over the full closing tail. The same argument applied to the write indicators gives $\mathbb{E}D = \sigma a \mathbb{E}\tau$.

Treat the initial request as if it was preceded by a private request. In every prefix ending with a private request, the number of write-to-private and private-to-write transitions is equal. Every private request is followed by a write with conditional probability a, so $\mathbb{E}Z = ap\mathbb{E}\tau$. The geometric closing tail has average length $1/p$, which proves $\mathbb{E}\tau = N + 1/p$. In the coverage identity (27), a public write has selection probability σ, a selected private request probability one, and an omitted helper probability zero, so the initial prefix contributes $\sigma a N W_N + apNJ$ to the expected numerator of the localized charge; dividing by its exact denominator gives $W_N + \rho J \le \bar{A}$. Partitioning all private work into that of selected and omitted private requests gives $p\Pi_N = apJ + \Gamma$, and substituting in $T_N = aW_N + p\Pi_N$ proves the last inequality.

For a worst-case-input budget, condition on the whole fixed input through τ and bound its conditional expected private work by P times its private count; averaging and dividing by $\mathbb{E}D$ gives $Pp/(\sigma a) = P\rho/a$. For a positionwise cap, bound only the selected private work by $K\mathbb{E}Z$, which gives $Kp/\sigma = K\rho$. The prefix's end point τ depends only on requests and not on the implementation's random tape, as required for both substitutions. □

Theorem E.4 (Localized charges and omitted private work)**.** *Under Lemma E.1's assumptions, set*

$$X = a\bar{A} - a^2 J/n + \Gamma, \qquad Y = \bar{A} + (1-\rho)J, \qquad Z_* = \min\{\Pi_N, Y\}.$$

Then

$$X, Z_* = \Omega(1), \qquad Y \le \bar{A}/\rho, \qquad \Phi(X, Z_*) = \Omega(H). \tag{37}$$

Consequently, for $N \ge \max\{4n, 1/p\}$,

$$\Phi\left(A_w + pQ, \min\{Q, A_w/\rho\}\right) = \Omega(H) \qquad \text{if } \Pi_N \le Q \text{ or } P \le Q, \tag{38}$$

$$\Phi\left(A_w + p^2 K, \min\{K, A_w/\rho\}\right) = \Omega(H) \qquad \text{under an individual private cap } K. \tag{39}$$

The cap statement also holds for an expected cap at each private position of each fixed input. All constants are uniform in p and in the budget. For finite costs, a fixed $c > 0$ gives the explicit helper deficit

$$\Gamma \ge \left[\frac{cH}{\log_2(2 + Z_*/H)} - a\bar{A} + \frac{a^2 J}{n}\right]_+. \tag{40}$$

The enlarged charge satisfies

$$A_w^+ \ge c, \qquad A_w^+ \log_2\left(2 + \min\{\Pi_N/A_w^+, \rho^{-1}\}\right) = \Omega(H) \tag{41}$$

on the same long horizons.

Proof. Lemma E.3 gives $T_N \le X$ and $W_N + J \le \bar{A} + (1-\rho)J = Y$; also $J \le \bar{A}/\rho$, so $Y \le \bar{A}/\rho$. Apply both parts of Lemma E.1 and use the bound with the smaller second argument. This proves (37). Since $\Gamma \le p\Pi_N$, one has $X \le \bar{A} + p\Pi_N$; under a cap, $\Pi_N, J \le K$ and $\Gamma \le Ke_N$. For $N \ge 1/p$, we have $c_N \le 2$ and $e_N \le 2p^2$, which proves (38)–(39) by monotonicity and constant-factor homogeneity.

The elementary inversion $\Phi(x, z) \ge cH \Rightarrow x \ge c'H/\log_2(2 + z/H)$ follows by considering $x \ge z$ and $x < z$; in the latter case it is the logarithmic inversion of Lemma C.1. Substitution of X gives (40). For the enlarged charge (41) put $\bar{F} = c_N A_w^+$. Its numerator includes all private probes, so $p\Pi_N \le \sigma a \bar{F}$, while $A_w \le A_w^+$. Thus $T_N \le 2\bar{F}$ and $\min\{\Pi_N, W_N + J\} \le \min\{\Pi_N, \bar{F}/\rho\}$. Use (33), noting $\Phi(x, y) \le Cx\log_2(2 + y/x)$ for all positive x, y, and the floor $T_N = \Omega(1)$; absorbing $1 \le c_N \le 2$ gives (41). □

E.3 Budget, cap, and three-class consequences

The lower bounds of this appendix come from three sources, and the corollaries below specialize them. Lemma E.1 (equivalently Theorem 3.4) gives the average-cost tradeoff $\Phi(T_N, \Pi_N) = \Omega(H)$. Theorem E.4 converts it into bounds on the localized charge on horizons $N \geq \max\{4n, 1/p\}$: (38) under average-input or worst-case-input budgets, (39) under individual caps, and the floor $A_w = \Omega(\rho)$, since (37) gives $\bar{A} \geq \rho Y \geq \rho Z_* = \Omega(\rho)$. Theorem E.26 gives the memory bound (48) from correctness alone, at every closed horizon. The bounds from Lemma E.1 and Theorem E.4 hold with disclosed or hidden private kinds and already for the public-write/private-read interface.

Corollary E.5 (Localized charge under an average private budget)**.** *Under the assumptions of Theorem E.4, for $N \geq \max\{4n, 1/p\}$, $\Pi_N \leq Q$ (in particular, $P \leq Q$) implies*

$$A_w = \Omega\left(\rho \max\left\{1, \frac{H}{\log_2(2 + pQ/H)}\right\}\right). \tag{42}$$

Proof. The terminal floor gives $A_w = \Omega(\rho)$. Set $z = A_w/\rho$. Equation (38) and $\Phi(x, y) \leq Cy \log_2(2 + x/y)$ imply $H \leq C' z \log_2(2 + pQ/z)$, since $\rho \leq 1$; uniform logarithmic inversion of Lemma C.1 proves (42). □

Corollary E.6 (Low worst-case-input budgets and individual caps)**.** *Under the assumptions of Theorem E.4, for $N \geq \max\{4n, 1/p\}$, there are fixed $c, C > 0$ such that, for $Q \geq CH$, a worst-case-input private budget at most Q forces*

$$A_w = \Omega\left(\max\left\{1, \frac{H}{\log_2(2 + Q/H)}\right\}\right) \quad \text{if } Q \leq \frac{cH}{p \log_2(2 + 1/p)}.$$

For an individual private cap K with $CH \leq K \leq cH/[p^2 \log_2(2 + 1/p)]$, the same bound holds with Q replaced by K. The first lower bound also applies to an average-input budget $\Pi_N \leq Q$ in its stated interval; it is stated for worst-case-input budgets because the matching construction (Corollary E.15) fits within such a budget.

Proof. Write $L_V = \max\{1, H/\log_2(2+V/H)\}$. Theorem E.4 and logarithmic inversion of Lemma C.1 give $A_w + pQ \geq c_1 L_Q$ under the budget and $A_w + p^2 K \geq c_1 L_K$ under the cap. In the budget interval $H/\log_2(2 + 1/p)$ is at most a fixed constant times L_Q; in the cap interval it is at most a fixed constant times L_K, since $\log_2(2 + K/H) = O(\log_2(2 + 1/p))$. Choose the interval constant c small enough to absorb the private term. The floors follow from the same inversion and $X = \Omega(1)$. □

Corollary E.7 (Private latency at a prescribed localized charge)**.** *Under the assumptions of Theorem E.4, there is a sufficiently small fixed $c_0 > 0$ such that, if $A_w \leq Q_w \leq c_0 F_p$ on a horizon $N \geq \max\{4n, 1/p\}$, then, with*

$$D_p(Q_w) = \max\left\{1, \frac{H}{\log_2(2 + Q_w/(pH))}\right\},$$

the omitted helpers contribute $\Gamma = \Omega(D_p(Q_w))$. Consequently

$$\Pi_N, P = \Omega(D_p(Q_w)/p), \qquad K = \Omega(D_p(Q_w)/p^2). \tag{43}$$

Here a bound on P or K means that any such worst-case-input budget or cap must have at least that size. If A_w is finite and Π_N is infinite, the finite bound on J forces Γ to be infinite and the bounds are immediate.

Proof. From (40), $Z_* \le 2Q_w/\rho$ and $\rho \ge p$ give $\Gamma \ge cH/\log_2(2 + 2Q_w/(\rho H)) - 2Q_w$. The positive first term is at least a fixed multiple of $D_p(Q_w)$; to show that it's also $\Omega(1)$, note that $\rho \ge 1/n$, $Q_w \le c_0 F_p \le H$, and $H = \Theta(\log_2 n)$ under the fixed polynomial memory gap; additionally, $D_p(Q_w) \ge cF_p$ when $Q_w \le c_0 F_p$. Taking c_0 small therefore gives $\Gamma \ge c' D_p(Q_w)$.

Helpers are private requests, so $\Gamma = \mathbb{E}[C_0 \mathbf{1}_G] \le \mathbb{E}[C_0 \mathbf{1}_{\mathcal{P}}] = p\Pi_N$, which gives $\Pi_N = \Omega(D_p(Q_w)/p)$; a worst-case-input private budget P bounds the average-input private cost, so $P \ge \Pi_N$. Under an individual cap K (or an expected cap at each private position), every helper costs at most K, so $\Gamma \le e_N K$ with $e_N = p^2 + ap/N \le 2p^2$, which gives $K = \Omega(D_p(Q_w)/p^2)$. □

Public reads interspersed with writes. The two-class formulas have a precise three-class accounting extension. Add public reads to the workload $\mathcal{W}_{W,R}$, with IID probabilities p, r, a for private reads, public reads, and public writes respectively, where $a, p > 0$ and $p+r+a = 1$. Compress each public-read run (possibly empty) and its following nonread into one macro. Its terminal is a public write with probability $\alpha = a/(a+p)$ and a private read with probability $\beta = p/(a+p)$. The implementation executes the read run before consulting the terminal request, as required by online execution. Fix a deterministic horizon of N macros and close it at the next private macro. Apply Definition E.2 to the macro terminals, including the complete macro whenever its terminal is selected. Thus A_w includes the preceding read runs of bridging writes and selected private requests.

All regular costs below refer to the first N macros. Let W_{mac} be the expected probes in complete public-write macros divided by αN, and U_{mac} the expected original private probes divided by βN. Let L be the expected probes in all preceding read runs divided by N. The terminal class is independent of its read run and the work already performed there, so the private-macro average is $U_{\mathrm{mac}} + L$ and

$$T_{\mathrm{mac}} = \alpha W_{\mathrm{mac}} + \beta(U_{\mathrm{mac}} + L).$$

Let J be the expected total cost of complete selected private macros in this prefix divided by $\alpha\beta N$. Let Γ and E be, respectively, the expected original-private probes and preceding-read-run probes of omitted private macros, each divided by N. Put

$$\rho = \beta + \alpha/n, \qquad \bar{A} = (1 + 1/(\beta N))A_w,$$
$$X = \alpha\bar{A} - \alpha^2 J/n + \Gamma + E, \qquad Z_* = \min\{U_{\mathrm{mac}} + L, \bar{A} + (1-\rho)J\}.$$

For the lower bound, retain the memory, word-size, and correctness assumptions of Lemma E.1 on the expanded workload. For computational privacy, retain its computational qualifications and additionally require efficiently sampleable request-class probabilities and polynomial exact expected probe counts for the expanded reference experiments, including their closing tails and forced terminals.

Lemma E.8 (Localized write bounds with public reads)**.** *With the macro accounting above, for every $N \ge 1$,*

$$W_{\mathrm{mac}} + \rho J \le \bar{A}, \qquad \Gamma + E = \beta(U_{\mathrm{mac}} + L - \alpha J), \qquad T_{\mathrm{mac}} \le X. \tag{44}$$

For the lower bound, additionally assume either unrestricted statistical trace privacy *or computational trace privacy with the qualifications above. Under these assumptions, if $N \ge 4n/\alpha$, then*

$$X/\alpha, Z_* = \Omega(1), \qquad \Phi(X/\alpha, Z_*) = \Omega(H). \tag{45}$$

If $\alpha \ge 1/2$, already $N \ge 4n$ suffices and X/α may be replaced by X. The constants are uniform in the class probabilities, and disclosed private kinds suffice.

Proof. The read-run length and terminal class of each macro are independent. Online execution makes the run's probe cost independent of that class as well, which justifies its contribution L to the private-macro average. After a public-write macro completes, its terminal is bridging with conditional probability $\sigma = \beta/(\beta + \alpha/n)$, independently of the macro's completed work. As in Lemma E.3, conditioning on each completed public-write macro shows that the expected cost of bridging macros through the closing private macro is σ times the expected cost of all public-write macros through that point. The same conditioning gives $\mathbb{E}D = \sigma\alpha(N + 1/\beta)$: the closing tail has expected length $1/\beta$ macros. Moreover, the selected cost in the first N macros is at most the total selected cost, including the tail; dividing by $\mathbb{E}D$ gives the first inequality in (44). Partitioning private-macro work into selected and omitted work gives the equality; substitution into T_{mac} gives the last inequality.

For the lower bound, group all probes by macro in the proof of Lemma E.1. Its terminal experiments now append either a private macro or a public-write macro followed by a private macro. Their expected complete costs are at most $4(U_{\mathrm{mac}} + L)$ and $4(W_{\mathrm{mac}} + J)$, respectively, by the same endpoint averaging and online conditioning. Take

$$\omega = \left\lfloor \frac{n}{1024\alpha} \right\rfloor, \qquad \ell_0 = \left\lceil \frac{C(m+1)}{\alpha} \right\rceil.$$

These windows fit before all chosen endpoints when $N \geq 4n/\alpha$, or when $\alpha \geq 1/2$ and $N \geq 4n$, and $\log_2(2 + \omega/\ell_0) = \Theta(H)$. The sources are public-write terminals of density α. Conditional on the source-count event of probability $g \geq 63/64$ (analogous to the event G in the proof of Lemma E.1), the expected number of possible later overwrites is at most $\alpha\omega/g + 1$; intervening public and private reads never overwrite source words. Hence the source-survival and encoding argument is unchanged. The paired experiments preserve all public requests and private kinds, so disclosed-kind privacy suffices. Their expanded logical lengths have unbounded support when $r > 0$, even though the number of macros is bounded. Since $a + p > 0$, each read run is almost surely finite. In the statistical case, condition on any realized input pair: its expanded inputs are finite and comparable, so unrestricted statistical privacy bounds their trace distance by $1/16$. Averaging over these pairs gives the same $1/16$ event-transfer loss in Lemma A.3. The closing tail enters only the cost accounting above.

The surviving-value argument of Lemma E.1 gives $T_{\mathrm{mac}} = \Omega(\alpha)$. Lemma A.6, with density α, therefore gives constant floors and the symmetric tradeoff with first argument T_{mac}/α and either second argument $U_{\mathrm{mac}} + L$ or $W_{\mathrm{mac}} + J$. Finally, (44) gives $T_{\mathrm{mac}} \leq X$ and $W_{\mathrm{mac}} + J \leq \bar{A} + (1-\rho)J$. Monotonicity yields (45); for $\alpha \geq 1/2$, constant-factor homogeneity allows X/α to be replaced by X. □

E.4 Matched write regimes and their resources

The write constructions use the localized charge A_w of Definition E.2. Throughout the remainder of this appendix, $a = 1 - p$, $\rho = p + a/n$, and $\sigma = p/\rho$. Initialization is charged separately. All constructions support the full API and hide private kinds; their security holds for adaptive logical requests, and formulas for A_w use the specified IID workload and its retrospective labels. Lemmas E.24–E.25 show that the implementations and their expected costs are well defined on the entire, possibly unbounded closing tail.

Theorem E.9 (Localized public write charge bounds with polynomially rare private requests)**.** *Fix $0 < \delta < 1/2$ and $0 < \eta < 1/2 - \delta$. Let $n^{-1/2+\delta} \leq p \leq n^{-\eta}$ and $CpH \leq Q_w \leq c_0$, where C may depend on δ, η. Let $D_p(Q_w) = \max\{1, H/\log_2(2 + Q_w/(pH))\}$.*

(a) Lower bound. *On $\mathcal{W}_{W,R}$ with $N \geq 4n$, every scheme with $A_w \leq Q_w$ has average private budget $Q = \Omega(D_p(Q_w)/p)$ and individual private cap $K = \Omega(D_p(Q_w)/p^2)$.*

(b) Upper bound. *Under the construction resources, with $m = \Theta(p^{-2}\log_2(1/Q_w))$ client words, $O(n)$ server cells and separate $O(n)$ initialization, a single hidden-request-kind construction has $A_w \leq Q_w$ on every closed horizon $N \geq 1$, private average- and worst-case-input budget $O(D_p(Q_w)/p)$, and individual cap $K = O(D_p(Q_w)/p^2)$.*

(c) Memory. *On every closed horizon $N \geq 1$, correctness and $A_w \leq Q_w \leq c_0$ alone imply*

$$m + 1 = \Omega\Big(\min\{n,\ p^{-2}\log_2(1/Q_w),\ N + p^{-1}\log_2(1/Q_w)\}\Big),$$

even if only physical server writes are counted (Theorem E.26). This means that the memory in (b) is optimal up to constants for $N \geq 4n$, as well as for any single implementation that meets the localized write budget on all horizons.

The terms in the memory lower bound correspond to storing the array, buffering across intervals without helpers, and retaining a request prefix of length N, with N chosen in advance. For short horizons, the last term can be the smallest one: with $p = n^{-1/4}$, $Q_w = n^{-1/8}$, and $N = 1$, Theorem E.27 uses $O(n^{1/4}\log_2 n)$ words.

Reserve $q = \Theta(m)$ client words with $n^\theta \leq q \leq n^{1-\varepsilon}$ for fixed positive θ, ε, and count the retained PRF keys and evaluation workspace as in Section 2.4; the additional table-state condition is stated in Theorem E.14. The constructions use $O(n)$ server cells and separately charged $O(n)$ initialization. The polynomial workspace permits linear archive rebuilding; this is a stronger memory hypothesis than the read construction needs. Put $J_H(Q) = \max\{1, H/\log_2(2 + Q/H)\}$ and recall $D_p(Q_w) = \max\{1, H/\log_2(2 + Q_w/(pH))\}$.

Theorem E.10 (Matched write regimes)**.** *Assume the construction resources with polynomial client workspace $n^\theta \leq q = \Theta(m) \leq n^{1-\varepsilon}$, $O(n)$ server cells, and separate $O(n)$ initialization. The following are constant-factor optima against the lower bounds of Lemma E.1, Theorem E.4, and Theorem E.26.*

(i) On $N \geq \max\{4n, \lceil 1/p \rceil\}$, the optimal A_w under a worst-case-input private budget Q is $\Theta(J_H(Q))$ for $CH \leq Q \leq cH/[p\log_2(2 + 1/p)]$; under an individual cap K, the optimal A_w is $\Theta(J_H(K))$ for $CH \leq K \leq cH/[p^2\log_2(2 + 1/p)]$. These assertions retain the table-state allowance of Theorem E.14.

(ii) If also $q \geq Cp^{-2}\log_2 n$ and $N \geq \max\{4n, n/p\}$, then for average-input or worst-case-input budgets $Q \geq CH/p$, the optimal A_w is $\Theta(p\max\{1, H/\log_2(2 + pQ/H)\})$.

(iii) In the regime of Theorem E.9, there exists a construction with $A_w \leq Q_w$ that simultaneously attains the optimal average-input and worst-case-input private budgets $\Theta(D_p(Q_w)/p)$ and the optimal individual cap $\Theta(D_p(Q_w)/p^2)$ on $N \geq 4n$ (Corollaries E.7 and E.16), with memory optimal up to constants; Corollary E.28 matches the complete finite-horizon memory curve (48).

In (iii), $H = \Theta(\log_2 n)$, so the threshold CpH agrees with the threshold $C_0 p\log_2 n$ of Corollary E.16. Parts (i) and (ii) leave a transition range open, which (iii) does not resolve by inversion, since constants would then enter an exponent.

Proof of Theorem E.9 and Theorem E.10. For (i), combine Corollaries E.6 and E.15. For (ii), combine Corollaries E.5 and E.23. For (iii) and Theorem E.9, use Corollaries E.7 and E.16 and Theorem E.26; Corollary E.28 gives the finite-horizon memory statement. □

Appendix E.10 also gives a prescribed-prefix construction matching the $N + p^{-1} \log_2(1/Q_w)$ memory branch in its stated regime. Theorem F.4 matches the total-cost lower bound with three request classes, using polynomial client memory; for the localized write charge, preceding public-read work must also be included, as in the macro accounting above.

E.5 A calendar built from refined consuming tables

A *calendar* is a schedule, indexed by request number, for rebuilding summaries of the request history and making them available to queries. In this subsection, we define a calendar that groups consecutive requests into nested intervals of geometrically increasing lengths and schedules a build for each interval; periodic snapshot builds consolidate older history into a bottom array. Each such build is a *job*. The calendar fixes its release, when its inputs are ready and the build may begin, and its public deadline, by which the build must finish for its result to enter the query view. The time between release and deadline is its *service window*: the job can proceed incrementally during this window while queries continue to use existing summaries.

A job's *scale* v measures the size of its build, and its work is *padded* to a fixed number of server probes determined by v, independently of the input. Jobs form recurring *types*: one for each interval scale and one for bottom snapshots, even if the bottom scale coincides with an interval scale. A type's *period* is the number of requests between successive releases of its jobs; in this calendar it equals the type's scale. A job's service window can be longer than its type's period. A separate service rule assigns rebuilding work to private helper requests, with public requests completing any work needed to meet the deadlines.

Background rebuilding in ORAM goes back to Ostrovsky and Shoup [OS97] and was developed further in [KLO12]; shuffled-input deamortization is developed in [AKLS21]. Our calendar additionally assigns service to specified private helpers and accounts for public deadline completion; these mixed-mode charges do not follow from a standard worst-case ORAM bound.

The consuming-table interface of Lemma B.2 cannot be used as an immutable dictionary: extraction retires the instance and lookups consume records. We therefore retain a separate immutable archive for each interval summary. This distinction also permits a rebuild to run concurrently with queries to its inputs.

Lemma E.11 (Archived summaries and resumable builds)**.** *Fix $\theta > 0$ and a workspace allowance $q \geq n^{\theta}$ satisfying the construction assumptions of Section 2.4. A capacity-s summary, for $q \leq s = O(n)$, can retain both an immutable canonical archive and a consuming lookup table, with $O(s)$ server cells, $O(\log_2 n)$ retained client words, and constant padded lookup cost. Its archive contains the largest-timestamp record for each key of its specified input, padded to capacity s. Given immutable source archives of total capacity $O(s)$, a fresh summary can be built using any reserved polynomial workspace $n^{\theta'}$ for fixed $\theta' > 0$, in $O_{\theta,\theta'}(s)$ probes, and can be paused between probes. Scanning an archive and building from it are private jointly with subsequent admissible queries to all live source tables.*

Proof. Read the source archives in fixed scans, sort by address and decreasing timestamp, retain the first record per address, and normalize to the public destination capacity. Write the result in canonical address order into a separate archive. Use a fresh oblivious random permutation to create the input to a new consuming table, then invoke Lemma B.2. Corollary B.7 supplies linear sorting, filtering, and random permutation with polynomial workspace; arrays fitting in that workspace are handled locally. All capacities and probe bounds are padded independently of the input.

The archive is read only by fixed scans, and the only retained outputs of processing it are logical canonical summaries and independently randomized new tables. Thus the continuation argument of

Lemma B.6 applies: no layout-dependent information is passed to later queries, and no existing table state is altered. Keeping the finite program state and private registers allows a build to pause; its scratch is disjoint from all live tables. All workspace, stashes, and the shared PRF evaluator are accounted for. □

The following service lemma applies to any collection of jobs satisfying its stated conditions; the calendar specified below provides one such collection.

Lemma E.12 (Private funding and public deadline completion)**.** *Suppose a padded job of scale $v \geq q$ has cv probes of work, is released after request r, and has a public deadline d with $v \leq d - r \leq 2v$. Its inputs remain available until completion. There are at most two jobs of each type, with geometrically growing scales $q = v_0 < \cdots < v_J \leq n$ and one additional scale in $[n, 2n]$; each type has period equal to its scale, and first releases no earlier than that scale. Put $L = J + 2$. For $0 < p \leq 1/2$ and $q \geq C/p^2$, there is a service schedule assigning job work to requests with the following properties:*

1. *Only private requests immediately preceded by a private request perform private maintenance.*
2. *On every input prefix, maintenance on its I private requests costs at most CLI/p; one private request costs at most CL/p^2.*
3. *Every job meets its deadline. Public maintenance per request is $O(n)$, and under IID modes its expected total through any stopping horizon τ with finite expected value is $O(e^{-c_0 p^2 q}\mathbb{E}\tau)$.*

The same expectation bound holds when each next mode is private with conditional probability at least p given the completed history.

Proof. Give each job an initially zero credit counter, capped at cv. On every private request add $\gamma = \lceil 64c/p \rceil$ credits to each unfinished job. If the preceding request was private, perform at most $\mu = \lceil 64c/p^2 \rceil$ of its remaining probes, spending one credit per probe. After a public request t, finish any job whose remaining work R_t exceeds $\gamma \max\{d - t - 1, 0\}$. A completed job discards its unused credits. The worst-case-prefix private budget follows because every spent credit was issued on a private request; at most $2L$ jobs are live. Their service quotas give the cap, and their geometric total size gives the public cap.

If the deadline request is public it finishes the job. Otherwise, after the last public request s the residual is at most $\gamma(d - s - 1)_+$, and the final private run supplies that many helper opportunities; each helper supplies at least $\min\{\gamma, R_t\}$ probes using its newly issued credits. If there is no public request after release, the private run has at least $d - r - 1$ helpers, and $cv \leq \gamma(d - r - 1)$ for the stated q. Thus all deadlines hold on every tape.

Condition on a job's release history and put $W_v = d - r$. Public forcing cannot occur before the midpoint of its window: $R_t \leq cv$ and the forcing rule imply $t > d - 1 - cv/\gamma \geq r + W_v/2$. In the first $\lfloor W_v/4 \rfloor$ positions, at least cv/γ private requests fully fund the residual work; in the next quarter, at least cv/μ disjoint pairs of consecutive private requests complete it. Each threshold is less than half its corresponding mean. Conditional exponential moments give the usual Bernoulli lower-tail bounds even under the conditional-probability hypothesis; the private pairs have conditional probability at least p^2. Hence the conditional probability of public completion is at most $Ce^{-c_0 p^2 v}$.

A type of period v has at most N/v releases in the first N requests, for N fixed in advance. Charge its entire possible future public completion to its release; the expected charge is $O(Ne^{-c_0 p^2 v})$, and summing over geometric scales gives $O(Ne^{-c_0 p^2 q})$. For a stopping horizon, $\{r \leq \tau\}$ is determined at release, so the same conditional bound applies to every released job, and on any run the release count is at most τ/v, proving the stated stopping version. This counts some jobs completing after τ, which only increases the bound. □

Our write calendar and query view. Fix integers $1 \le q \le n$ and $B \ge 2$, and put $v_i = qB^i$ for $0 \le i \le J$, where $v_J \le n < v_{J+1}$, and $v_* = q\lceil n/q \rceil$. The scale-v_i intervals are the aligned intervals $((k-1)v_i, kv_i]$, for integers $k \ge 1$; for $i \ge 1$, each is the union of B consecutive scale-v_{i-1} intervals, its *children*. Request t appends one record, possibly a dummy, at position t of the request history. A public write appends its supplied value without searching; every other request searches for its key's current value and then appends that value or, for a private write, its supplied value. Private reads and writes perform identical padded searches. A *summary* retains the latest timestamped record of each key in its interval and consists, as in Lemma E.11, of an immutable archive and a consuming lookup table.

Each request performs its search and append, services jobs, publishes due outputs, and then releases new jobs, so a job released after request r and published after d is serviced during requests $r+1, \dots, d$. The summary of a scale-q interval ending at e is built from the client buffer holding its appends, kept until publication; the build is released after e and published after $e + 2q$. For $i \ge 1$, the summary of a scale-v_i interval ending at e is built from its children's immutable archives; the build is released after $e + 2v_{i-1}$, when the last child publishes, and published after $e + 2v_i$, so its service window has length $2(v_i - v_{i-1}) \in [v_i, 2v_i]$. At each positive multiple b of v_*, release a bottom snapshot build with cutoff b: freeze the current *query view* (information available to queries: client buffers, current published intervals, and the current snapshot), and build from it a representation of all values at b; the snapshot is published after $b + 2v_*$. Just like interval summaries, the snapshot contains an immutable archive and a consuming lookup table. The initial bottom snapshot is the input array, with cutoff zero.

A search uses the consuming lookup tables. It examines records from newest to oldest; first in the client buffers, then in the published intervals still in the query view, ordered from newest to oldest, and finally in the bottom snapshot. After finding the requested item, it performs dummy lookups for the remaining tables in the query view. Outputs due at the same request publish in increasing scale order, with the bottom snapshot being published last. A job that finishes early still waits for its scheduled publication. When a parent is published, its children are removed from the query view. When the bottom snapshot is published, the old bottom snapshot and the intervals ending at or before the new snapshot cutoff are removed from the query view.

Build inputs are retained as follows. A child's archive is reserved until its parent's scheduled publication, even if the child was removed from the query view earlier by a snapshot. A bottom snapshot build pins the immutable archives in its frozen view and copies the bounded client buffers into its own reserved workspace. An archive is removed only after it has left the query view and every parent reservation and snapshot pin on it has ended; a lookup table can be discarded as soon as it leaves the query view.

Lemma E.13 (Calendar coverage, nonrecurrence, and storage)**.** *Suppose that every job in the above calendar completes by its scheduled publication and that its archives and lookup tables are built correctly. At every request boundary the following invariants hold.*

(i) The search procedure returns the current value of every key. An interval crossing the bottom snapshot cutoff can remain searchable without returning a stale value.

(ii) Each real key is queried at most once in any lookup-table instance, including after a miss. The request's append following the search protects that instance until it leaves the query view.

(iii) All consuming tables (in the query view), immutable archives (in the query view, reserved by a build or pinned by a snapshot), unfinished outputs, and server scratch together occupy $O(n)$ cells. There are $O(B(J+2))$ query tables and at most two live jobs of each scale, including the bottom scale. Client buffers and their frozen copies occupy $O(q)$ words.

The same assertions hold when public reads append dummy slots and perform no search (this can happen if public reads are served by a separate structure).

Proof. We describe the logical coverage without consumption, verify that consumption preserves it, and finally count all retained objects.

Coverage and publications. At a boundary t, let b be the published bottom snapshot's cutoff. Every append after b belongs to a retained client buffer or a maximal published interval: initially this follows from the bottom snapshot and the current buffer; a completed base summary replaces its buffer; a parent summary replaces its children and represents precisely their combined interval, the reserved archives supplying all of those children's records regardless of earlier consuming lookups or query-view removals. Every new summary retains the latest record of each key in its interval, so replacing children by their parent preserves chronological lookup.

A bottom build uses its frozen view, including the archives and buffer copies, to obtain the latest value of each key as of its cutoff; later lookups cannot alter those inputs. When this snapshot publishes, all appends after its cutoff remain in buffers or intervals, and intervals wholly before the cutoff can be discarded. If $I = (s, e]$ straddles the cutoff b and contains a record for x at time $u \le b$, then every later append to x through b also lies in I, so its summary contains a value at least as recent as the bottom snapshot's value for x, and newer still if it contains an append after b. This justifies searching the straddling interval before the bottom snapshot. An interval ending at $e \le b$ publishes by $e + 2v_i \le b + 2v_*$, so an obsolete interval cannot enter the query view after that bottom snapshot; the prescribed publication order resolves the case when both the interval and the snapshot are published at the end of the same request.

Consumption and protection. Suppose a real (non-dummy) search for x reaches a table instance whose interval ends at e, or whose bottom cutoff is e, during request t. Then $t > e$, and the request appends a record for x at time t, even on a miss. As long as the instance remains searchable, this append or a later record for x is represented in an earlier-searched component, and merging younger components preserves that record. If a parent summary incorporates both the protected instance's interval and the append, it removes the protected instance from the query view; if a bottom cutoff reaches the append, it has also passed the older interval's end and removes that instance, and replacing a bottom snapshot always retires the old bottom snapshot instance. Hence neither type of publication can remove the protection while leaving the protected instance searchable. Any later consumption of a protecting record creates a still later append, so induction preserves the claim, and the same induction shows that consumed records retained in archives are harmless: they are superseded by a later append until a summary incorporates that append. Consequently lookups are correct and no real key recurs in a table instance. Public writes supply newer records directly; dummy appends neither consume records nor remove protection.

Reservations and storage. For $i < J$, a scale-v_i interval ending at e belongs to a parent ending at $e' \le e + v_{i+1} - v_i$, whose regular reservation ends at $e' + 2v_{i+1} < e + 3v_{i+1}$. Since consecutive scale-v_i intervals end v_i requests apart, there are $O(B)$ such archives at a time, including unpublished outputs and archives whose tables have retired. At the largest interval scale, choose the first bottom cutoff $b \ge e$; it satisfies $b < e + v_*$, and its publication removes the interval by $e + 3v_*$, so there are $O(1 + v_*/v_J) = O(B)$ top-scale archives under regular retention. Their total capacity is bounded by

$$O\left(\sum_{i=0}^{J-1} Bv_i + v_* + v_J\right) = O(n),$$

using $\sum_{i=0}^{J-1} Bv_i = \sum_{i=1}^{J} v_i \le 2v_J$ and $n \le v_* < n + q$. The published bottom snapshot adds $O(n)$ cells.

Every job is live for at most twice its release period, and publications precede releases at the same request boundary, so there are at most two jobs of each scale. Each bottom job pins one query view of total capacity $O(n)$, so the archives retained solely by the at most two bottom pins add $O(n)$ cells. Those pins do not retain client-side stashes of the consuming lookup tables after a table leaves the query view. Disjoint build outputs and scratch cost $O(\sum_{i=0}^{J} v_i + v_*) = O(n)$, with the constant absorbing the two simultaneous jobs per scale and their linear-space sorting procedures; this also bounds server storage during simultaneous publications.

There are at most two completed length-q buffers awaiting publication and one current buffer. The two bottom jobs each freeze at most this many buffers, so their total local storage is $O(q)$. Each retained query table is associated with a retained summary, giving $O(B(J+2))$ table stashes. The remaining per-job workspace is allocated in the proof of Theorem E.14. □

An interval crossing a bottom cutoff. For example, take $B = 2$, a scale $s = q2^j$, and $n = 5s$, so $v_* = 5s$ and the largest interval scale is $4s$. Three relevant jobs have the following times:

Object	Release	Publication
Bottom snapshot with cutoff $5s$	$5s$	$15s$
Interval $(4s, 6s]$	$8s$	$10s$
Parent interval $(4s, 8s]$	$12s$	$16s$

At time $15s$, the new bottom snapshot removes intervals ending at or before $5s$, but keeps $(4s, 6s]$. Suppose x is updated at $4s+1$ and $5s$, and y at $5s+1$, with no other updates to these keys. This interval's archive retains the latter value of x, agreeing with the bottom snapshot, and the value of y remains newer than the bottom snapshot. At $16s$, the parent replaces its two children and retains their latest records, including that update to y. If a child's lookup copy was previously consumed, the corresponding later append provides the protection in part (ii); the immutable child archive remains available for the parent build until its publication.

Theorem E.14 (Computational write calendars)**.** *Fix $\theta, \varepsilon > 0$ with $\theta < 1 - \varepsilon$ and choose $n^\theta \le q \le n^{1-\varepsilon}$. Assume $w \ge C \log_2 n$ and reserve a constant fraction of qw bits for the retained PRF keys and the entire shared evaluation workspace. For an integer $2 \le B \le \lfloor n/q \rfloor$, put $L = \lfloor \log_B(n/q) \rfloor + 2$, and suppose $BL \log_2 n \le cq$ for a sufficiently small fixed allocation constant. There is a computationally private full-API calendar with $O(q)$ client words, $O(n)$ server words, and $O(n)$ separate initialization. Let $\lambda = \Theta(BL)$ be a uniform upper bound on its lookup cost. It has two schedules:*

$$P, K = O(BL), \qquad A_w = O(L + \rho BL) \qquad \textit{(eager service)}, \tag{46}$$

$$P \le \lambda + CL/p, \qquad K \le \lambda + CL/p^2, \qquad A_w \le \rho\lambda + Ce^{-c_0 p^2 q}/\sigma \qquad \textit{(private service)}. \tag{47}$$

For these constructions, the private P-budget additionally holds pointwise: on every full-API input prefix and every random tape, total private probes are at most P times the private count. The cap K bounds each private request on every tape. The second schedule requires $q \ge C/p^2$; the first has no such requirement. Both charge formulas hold for every deterministic $N \ge 1$, closed at its next private request, with no startup or closing-tail term. The private-service schedule gives the sharper $A_w \le C(pBL + e^{-c_0 p^2 q})$ whenever $np \ge 1$.

Proof. Calendar invariants and workspace. Use the calendar just specified, with $L = J + 2$. Lemma E.13 supplies coverage, nonrecurrence, and the complete $O(n)$ server-space accounting whenever jobs finish on time; the schedules below ensure those deadlines on every tape. The lemma also bounds the number of live query tables by $O(BL)$, the number of concurrent jobs by $O(L)$,

and all client buffers and their frozen copies by $O(q)$. Allocate $q/O(L)$ private workspace to each live build; this is at least $n^{\theta/2}$ for sufficiently large n, so Lemma E.11 gives a linear build bound with a constant independent of B, p, N. The $O(BL \log_2 n)$ client stashes, job state, buffers, and counters fit reserved portions of $O(q)$ words. Only one PRF evaluation is active at a time; its shared workspace is counted separately within that allowance. Pinned archives need no retained lookup stashes after their tables leave the query view. Initialization applies the summary-building procedure of Lemma E.11 to the initial array, producing the initial bottom snapshot's archive and consuming lookup table.

Service and costs. For eager service give each live job a sufficiently large constant probe quota per request. Its service window is at least its scale, so it finishes by its fixed deadline, and maintenance costs $O(L)$ on each request. Public writes skip lookup; every selected private request pays at most $\lambda + O(L)$. The expected number of selected private operations divided by the expected number of bridging writes is exactly $p/\sigma = \rho$, and a public write's maintenance is bounded by CL pointwise, including after its retrospective selection. This proves (46).

For private service apply Lemma E.12. Selected private requests follow public writes, so none performs private maintenance; each pays at most λ lookup probes, contributing at most $\rho\lambda$ to A_w. Public maintenance, even without thinning, has expectation $O(e^{-c_0 p^2 q}\mathbb{E}\tau)$; divide by $\mathbb{E}D = \sigma a \mathbb{E}\tau$ and use $a \geq 1/2$ to obtain the last inequality of (47). If $np \geq 1$, then $\sigma \geq 1/2$ and $\rho \leq 2p$. The credit funding mechanism gives the worst-case-input private budget, and the credit spending quota gives the individual cap.

Privacy and bounded resources. Archives expose only fixed scans and canonical logical outputs. Each new lookup table gets independent fresh randomness and has the nonrecurrence property already proved, so the reactive table simulators and Lemma E.11's continuation privacy proof compose, including under adaptively chosen logical requests. Schedules, publication times, credits, capacities, and padded work depend only on disclosed modes and counters. Convention B.3 preserves the stated storage and private bounds on every tape with negligible correctness and privacy error. These pointwise guarantees must also hold throughout the closing tail, whose length is unbounded. Lemma E.24 supplies bounded clocks and renewable key domains for that purpose, while Lemma E.25 justifies taking expectations over the full tail. □

E.6 Tuning the calendar

Corollary E.15 (Low-budget write schedules)**.** *Fix the polynomial memory allowance of Theorem E.14. Let $H = \log_2(2 + n/q)$ and $J_H(Q) = \max\{1, H/\log_2(2 + Q/H)\}$. For $Q \geq CH$, the eager calendar can be tuned so $P, K \leq Q$ and*

$$A_w = O(\max\{J_H(Q), \max\{1, H/\log_2(2 + 1/p)\}\}).$$

In particular $A_w = O(J_H(Q))$ throughout $CH \leq Q \leq cH/[p^2 \log_2(2 + 1/p)]$. Memory has no additional dependence on p.

Proof. Choose

$$B = \min\left\{\lfloor n/q \rfloor, \lfloor n^{\theta/2} \rfloor, \lfloor 1/p \rfloor, \max\{2, \lfloor Q/(C_1 H) \rfloor\}\right\},$$

with fixed C_1 sufficiently large. Then $BL \log_2 n = o(q)$, $BL \leq Q/C_2$, and $pB \leq 1$. Eager service has $P, K \leq Q$ and $A_w = O(L)$, since also $B/n \leq 1/q$. Before clipping, elementary logarithmic inversion gives $L = O(J_H(Q))$. Clipping at either polynomial memory limit makes $L = O(1)$, and clipping at $1/p$ makes $L = O(\max\{1, H/\log_2(2 + 1/p)\})$. In the stated range of Q, $\log_2(2 + Q/H) = O(\log_2(2 + 1/p))$, which proves the specialization. □

Corollary E.16 (A small localized write charge with bounded private latency)**.** *Fix* $0 < \delta < 1/2$ *and* $0 < \eta < 1/2 - \delta$*. Suppose*

$$n^{-1/2+\delta} \le p \le n^{-\eta}, \qquad C_0 p \log_2 n \le Q_w \le Q_{w,0},$$

where $Q_{w,0} > 0$ *is a sufficiently small constant and* C_0 *sufficiently large. Set*

$$q = \left\lceil C_q p^{-2} \log_2(1/Q_w) \right\rceil, \qquad H = \log_2(2 + n/q), \qquad D_p(Q_w) = \max\left\{1, \frac{H}{\log_2(2 + Q_w/(pH))}\right\}.$$

Assume the counted PRF resources of Theorem E.14, with $m = \Theta(q)$ *client words. There exists a hidden-kind computational family that has* $O(n)$ *server words,* $O(n)$ *separate initialization, and, for every closed deterministic horizon* $N \ge 1$*,*

$$A_w \le Q_w, \qquad P = O(D_p(Q_w)/p), \qquad K = O(D_p(Q_w)/p^2).$$

The average-input private cost is at most P*. At horizons* $N \ge 4n$*, Theorem E.26 makes this memory order optimal for the same localized write charge target, and Corollary E.7 makes the budget and cap optimal on the specified workload.*

Proof. The chosen q lies between fixed positive polynomial powers of n, and $q \le n^{1-\delta}$ for sufficiently large n; hence $H = \Theta(\log_2 n)$. Choose

$$B = \min\left\{\lfloor n/q \rfloor, \lfloor 1/p \rfloor, \max\{2, \lfloor Q_w/(C_A p H) \rfloor\}\right\}.$$

Take C_A, C_0, C_q sufficiently large. Since $np \ge 1$, Theorem E.14 gives $A_w \le C(pBL + e^{-c_0 p^2 q})$, and we bound each term by $Q_w/2$.

Size of B *and* L*.* As $Q_w/H = o(1)$, we have $Q_w/(C_A p H) < 1/p$, so the cap $\lfloor 1/p \rfloor$ never binds, though $B \le 1/p$ still holds. If $B = \lfloor n/q \rfloor$, then $L = O(1)$. Otherwise $B = \max\{2, \lfloor Q_w/(C_A p H) \rfloor\}$, and logarithmic inversion gives $L = O(D_p(Q_w))$. Always $L \le H + 2 \le 3H$.

Charge. Either $B \le Q_w/(C_A p H)$, and then $CpBL \le 3CQ_w/C_A \le Q_w/2$; or $B = 2$, and then $CpBL = O(p \log_2 n) \le Q_w/2$ by $Q_w \ge C_0 p \log_2 n$. The choice of q gives $e^{-c_0 p^2 q} \le Q_w^{c_0 C_q}$, so $Ce^{-c_0 p^2 q} \le Q_w/2$. Hence $A_w \le Q_w$.

Hypotheses and costs. Since $B \le 1/p$ and $p \le n^{-\eta}$, $BL \log_2 n = O(p^{-1} \log_2^2 n) = o(q)$, so all stashes fit. Finally $\lambda = \Theta(BL) = O(L/p)$, so (47) gives the asserted budget $P = O(D_p(Q_w)/p)$ and cap $K = O(D_p(Q_w)/p^2)$.

The pointwise worst-case-input budget bounds the average-input private cost. The memory lower bound at $N \ge 4n$ is $\Omega(\min\{n, p^{-2} \log_2(1/Q_w)\}) = \Omega(q)$ in this regime. □

E.7 Average write costs and a buffering example

The charge A_w includes preparation performed by selected private requests, so a lower bound on it need not hold for the average cost W_N of a public write. The following elementary example separates the two measures without cryptography.

Proposition E.17 (Buffering public writes)**.** *For every integer* $1 \le b \le n$*, there is a perfectly trace-private full-API implementation with* $O(b)$ *client words,* n *server words, and exact correctness. On* $\mathcal{W}_{W,R}$*, at every deterministic* $N \ge 1$*,*

$$\Pi_N = P = K = 2n, \qquad W_N \le 2n(1-p)^b, \qquad A_w \ge 2n\rho.$$

Here P and K are valid worst-case-input and individual private bounds, and initialization is separate. In particular, at constant p, a buffer of $O(\log_2 n)$ words makes W_N inverse-polynomially small while $A_w = \Omega(n)$. More generally, on the three-class workload of Theorem 3.5 with $a, p > 0$, the same implementation satisfies, at every deterministic $N \geq 1$,

$$\Pi_N = 2n, \qquad R_N = 1 \quad (r > 0), \qquad W_N \leq 2n \left(\frac{a}{a+p}\right)^b.$$

In particular, at constant $p, r, a > 0$, a buffer of $O(\log_2 n)$ words makes W_N inverse-polynomially small while $R_N = 1$.

Proof. Keep a canonical server array and a client list of at most b pending public writes, in chronological order. A public write appends locally; if the list is full, first read and overwrite every server cell in fixed address order, applying the list, and clear it. A public read always probes its disclosed canonical address and overrides the answer with the newest matching buffered value. Every private request scans and overwrites all n cells, applies the buffered writes followed by its own operation, returns its answer if appropriate, and clears the list; this costs exactly $2n$ private probes for either private kind.

Only disclosed modes and public requests determine buffer occupancy and public flushes, and every private trace is the same full scan, so the trace depends only on the permitted leakage. On the workload, a public write can flush only if at least b consecutive public writes precede it; conditional on the current request being public, this has probability at most $(1-p)^b$, proving the bound on W_N. Every selected private request costs $2n$, and Lemma E.3 gives $\mathbb{E}Z/\mathbb{E}D = \rho$, including the closing tail, so its contribution alone gives $A_w \geq 2n\rho$.

With three request classes, public reads leave the buffer unchanged and cost one probe each. A public write can flush only if the b preceding requests other than public reads exist and are all public writes. Given the positions of these requests, their classes are independent with public-write probability $a/(a+p)$, so conditional on the current request being a public write, a flush has probability at most $(a/(a+p))^b$, proving the bound on W_N. □

The private-service write calendar of Theorem E.14 provides smaller private costs while keeping the average cost of public writes very low.

Corollary E.18 (Average costs of the private-service calendar)**.** *Fix $d > 0$. Under Theorem E.14's counted resources, suppose $q \geq C_d p^{-2} \log_2 n$, $2 \leq B \leq \min\{\lfloor n/q \rfloor, 1/p\}$, and $BL \log_2 n \leq cq$, with $L = \lfloor \log_B(n/q) \rfloor + 2$. On $\mathcal{W}_{W,R}$ and every polynomial horizon $N \geq 4n$, the private-service calendar satisfies*

$$W_N = O(n^{-d}), \qquad T_N = \Theta(L), \qquad \Pi_N = \Theta(L/p),$$
$$P = O(L/p), \qquad K = O(L/p^2), \qquad A_w = \Theta(pBL).$$

The constants may depend on d and the fixed memory exponents. The theorem's linear server space and initialization bounds are unchanged.

Proof. A public write makes no lookup; its only probes complete calendar jobs. Lemma E.12 bounds their expected total by $CNe^{-c_0 p^2 q}$, and since $a \geq 1/2$, division by aN gives $W_N \leq C' e^{-c_0 p^2 q} = O(n^{-d})$ for large C_d. The private bounds in (47) and $B \leq 1/p$ imply $P = O(L/p)$, $K = O(L/p^2)$, and $T_N = O(L)$.

A private request following a public write performs only its padded lookup, so the conditional cost J is at most $\lambda = \Theta(BL)$. Lemma E.1 gives $\Phi(T_N, W_N + J) = \Omega(H)$. Since $\Phi(x, y) \leq Cx \log_2(2 + y/x)$, logarithmic inversion yields $T_N = \Omega(H/\log_2(2 + BL/H)) = \Omega(L)$, where $H = \Theta(L \log_2 B)$ because

$q = \Theta(m)$ and $2 \le B \le n/q$. Subtracting $aW_N = o(1)$ from $T_N = aW_N + p\Pi_N$ gives $\Pi_N = \Theta(L/p)$. The memory conditions imply $np = \omega(1)$, hence $\rho = \Theta(p)$ and $N + 1/p = \Theta(N)$.

For the lower bound on A_w, fix a request boundary $u \ge 2n$. For each $0 \le i < L-2$, the $2(B-1)$ scale-v_i intervals with endpoints

$$u - 2v_{i+1} < e \le u - 2v_i$$

have published by u. A larger interval containing one of them, or a bottom snapshot whose cutoff reaches its endpoint, publishes no earlier than $e + 2v_{i+1} > u$. Thus all these tables remain in the query view. There are $L - 2 \ge 1$ such scales, so every search after boundary $2n$ probes $\Omega(BL)$ live tables. Each of the final $N - 2n \ge N/2$ positions of the prefix is a selected private request with probability ap, so these positions contribute $\Omega(apNBL)$ expected selected-private probes. Dividing by $\mathbb{E}D = \sigma a(N + 1/p)$ from Lemma E.3 gives $A_w = \Omega(pBL)$. The matching upper bound follows from (47), increasing C_d to absorb the exponentially small public contribution into pBL. □

Corollary E.19 (Private budget needed for small average write cost)**.** *Under the lower-bound assumptions of Lemma E.1, there are constants $c_0, c_1 > 0$, independent of p, such that*

$$W_N \le c_0 F_p \quad \Longrightarrow \quad \Pi_N \ge c_1 F_p/p.$$

This holds with disclosed or hidden private kinds. Conversely, for every fixed $d > 0$, under the counted construction assumptions of Corollary E.18, including $q \ge C_d p^{-2} \log_2 n$ and polynomial client workspace, there is a hidden-kind full-API family satisfying

$$W_N = O(n^{-d}), \qquad \Pi_N = \Theta(F_p/p), \qquad P = O(F_p/p)$$

on every prescribed polynomial horizon $N \ge 4n$, with $O(n)$ server words and separate $O(n)$ initialization. Thus the private-budget scale necessary for average write cost below a sufficiently small multiple of F_p is sufficient, up to constants, for inverse-polynomial average write cost in this resource regime.

Proof. Since $\Pi_N \le T_N/p$ and $\Phi(T_N, \Pi_N) \le CT_N \log_2(2 + \Pi_N/T_N)$, Lemma E.1 and its constant floor give $T_N \ge cF_p$. For $c_0 \le c/2$, subtraction gives $p\Pi_N = T_N - (1-p)W_N \ge cF_p/2$.

For the upper bound choose $B = \min\{\lfloor n/q \rfloor, \lfloor q^{1/4} \rfloor, \lfloor 1/p \rfloor\}$. For sufficiently large n, $B \ge 2$, $BL \log_2 n = o(q)$, and $L = \lfloor \log_B(n/q) \rfloor + 2 = O(F_p)$: probability clipping at $1/p$ gives the denominator $\log_2(2 + 1/p)$ in the definition of F_p, and either polynomial resource clip gives constant depth. Corollary E.18 gives $W_N = O(n^{-d})$ and $P = O(F_p/p)$, and the lower bound just proved applies for large n and supplies $\Pi_N = \Omega(F_p/p)$. □

E.8 The high-private-budget localized-write frontier

The next construction uses the authority hierarchy of Appendix B directly.

Lemma E.20 (A public-reset compiler for a worst-case-input private budget)**.** *Suppose an implementation with padded request probes has a nonnegative, initially zero potential Υ, with private probes plus $\Delta\Upsilon$ at most α, and public $\Delta\Upsilon$ at most $\beta > 0$. Here $\Delta\Upsilon$ denotes the change in potential after processing the request. A public reset preserves the logical state, sets $\Upsilon = 0$, and costs at most R_*. Choose $P = \alpha + 2\beta/p$ and $s = 2\beta/p$. Maintain $C = PI - V$, where I is the number of private requests so far, and V is their total number of probes. After a public request, reset if $\Upsilon > C + s$, without changing C. Then every prefix satisfies $V \le PI$. Under IID modes with $0 < p \le 1/2$, the expected total number of resets over the entire execution is $O(1)$, so their expected work is $O(R_*)$, also on any stopping prefix. The reset decisions preserve privacy when all padded costs and potentials depend only on disclosed counters.*

Proof. Maintain $C \geq 0$ and $\Upsilon \leq C + s$. A private request satisfies $C_{\text{new}} - \Upsilon_{\text{new}} \geq C_{\text{old}} - \Upsilon_{\text{old}} + s \geq 0$; a public request either preserves the invariant or resets Υ. Thus the claimed budget holds without initial private credit.

Let $S_t = \beta(t - I_t) - sI_t$, and let D_t be the total potential discarded by resets through time t. Each public request increases $\Upsilon - C$ by at most β, while each private request decreases it by at least s. Summing these changes and accounting for resets gives $\Upsilon - C \leq S_t - D_t$. Immediately before the kth reset, $\Upsilon - C > s$. Each earlier reset discarded more than s, since it occurred when $\Upsilon > C + s$ and $C \geq 0$. Thus $D_t > (k-1)s$ for $k > 1$ (and $D_t = 0$ for $k = 1$), so $S_t > ks$. Set $\zeta = p/(4\beta)$. The increment $S_{t+1} - S_t$ equals β with probability $1 - p$ (a public request) and $-s$ with probability p (a private request). Since $\zeta\beta = p/4$ and $\zeta s = 1/2$, we have

$$\mathbb{E}\left[e^{\zeta(S_{t+1} - S_t)}\right] = (1-p)e^{p/4} + pe^{-1/2} \leq (1-p)(1 + p/3) + 2p/3 = 1 - p^2/3 \leq 1,$$

where we used $e^{p/4} \leq 1 + p/3$ for $0 < p \leq 1/2$ and $e^{-1/2} < 2/3$. Hence $e^{\zeta S_t}$ is a nonnegative supermartingale, and Ville's maximal inequality [HRMS20, Lemma 1] gives $\Pr\{Z \geq k\} \leq e^{-\zeta ks} = e^{-k/2}$ for the total reset count Z; summing proves $\mathbb{E}Z = O(1)$. □

Lemma E.21 (A bounded credit bank)**.** *Suppose additionally that the potential in Lemma E.20 is bounded pointwise by a public V_*. Round α, β upward to positive integers, put $s = \lceil 2\beta/p \rceil$ and $P = \alpha + s$, and choose an integer $h \geq 1$. The compiler of Lemma E.20 can be modified to store a bounded credit counter $0 \leq C \leq C_* := \lceil V_* \rceil + h$, while retaining the private P-budget on every execution, starting from $C = 0$. On any stopping prefix τ with finite expected value, under IID modes, its reset count satisfies $\mathbb{E}Z_\tau = O(1 + e^{-ph/(4\beta)}\mathbb{E}\tau)$. When all parameters are polynomially bounded, the bank occupies $O(\log_2 n)$ bits. Its decisions depend only on the same disclosed padded costs and counters as the original compiler.*

Proof. On a private request of cost v, replace the update $C \leftarrow C + P - v$ by $C \leftarrow \min\{C_*, C + P - v\}$; retain the public reset rule $\Upsilon > C + s$. Before clipping, a private update gives $C_{\text{new}} \geq \Upsilon_{\text{new}}$ by the original invariant, and clipping preserves this inequality because $C_* \geq V_*$. Thus $C \geq 0$ and $\Upsilon \leq C + s$ hold on every tape. If D is the total discarded credit, then $C = PI - V - D$, so $V \leq PI$. Note that the unbounded counts I, V, and D are only analysis variables not stored by the construction.

Rounding s upward only decreases the factor $e^{-\zeta s}$, so the exponential supermartingale bound remains valid. Before the first clipping, the original reset-count bound applies. Immediately after a clipping, $C - \Upsilon \geq h$; until the next clipping or reset this difference increases by at least s on a private request and decreases by at most β on a public request. Ville's inequality [HRMS20, Lemma 1] applied to the exponential supermartingale of Lemma E.20, with $\zeta = p/(4\beta)$, bounds the conditional probability of a first reset before the next clipping by $e^{-\zeta(h+s)}$. After such a reset, $\Upsilon = 0$ and $C \geq 0$, and the original argument bounds the expected number of further resets before the next clipping by an absolute constant. Each clipping consequently starts an interval with conditional expected reset count $O(e^{-\zeta h})$. Charge that count to the clipping, even if the resets occur after τ; whether the clipping lies in the stopped prefix is determined at its completion, and there are at most τ clippings. Conditional expectation and summation give the claimed bound. □

Theorem E.22 (A deferred authority with a public-write FIFO)**.** *Fix $\theta, \varepsilon > 0$ and let*

$$\max\{n^\theta, C_G p^{-2} \log_2 n\} \leq q \leq n^{1-\varepsilon}, \qquad 0 < p \leq 1/2,$$

with sufficiently large fixed C_G. Assume the word-size and counted PRF-workspace conditions of Theorem E.14; the calendar's $BL \log_2 n$ stash condition is not needed here, since the authority has

only $O(L)$ active tables. For $2 \le B \le \lceil n/q \rceil$ and $L = \lceil \log_B(n/q) \rceil$, there is a full-API implementation with hidden-kind adaptive computational privacy, $O(q)$ client words, $O(n)$ server words, and $O(n)$ separately paid initialization, such that

$$P = O(BL/p), \qquad A_w \le C\left(pL + \frac{n}{N}\right)$$

for every prescribed polynomial deterministic horizon $N \ge n$, closed at its next private request. The private P-budget additionally holds pointwise: it bounds the average over private requests for every execution on a prefix.

Proof. Deferred engine. Use only the authority of Section B.3, with growth factor B. Each access searches the client buffer and all active authority tables, then appends. Each active-table query retains its prescribed padding. Let $\lambda = \Theta(L)$ be a uniform upper bound on the lookup probe cost. Allow regular private carries to be deferred for at most $2G$ accesses, where $G = \lceil c_G p^{-2} \log_2 n \rceil$ and $q \ge 8G + 8$, and increase the buffer slack Δ to $2G + 2$. The proof of (24) is unchanged, since the remaining buffer suffix is still shorter than q and every batch contains exactly q slots; in particular, each carry costs $O(n)$.

We amortize carries with credits. A level-i carry runs once per qB^{i-1} appends and costs at most its fixed padded bound CS_i, so each append deposits $CS_i/(qB^{i-1}) \le CB$ credits for every level i, in total $d = O(BL)$ per append. Let $\Psi \ge 0$ be the unspent balance; then every access satisfies authority probes $+ \Delta\Psi \le e := \lambda + d$. Each carry is paid by the deposits of its own period, so Ψ holds only the deposits of each level's current, incomplete period, at most $\sum_i CS_i = O(n)$, plus, from the at most $2G + 2$ appends whose carries are deferred, at most $(2G + 2)e = O(n)$; hence $\Psi = O(n)$. Here $qe = O(n)$, using $B\lceil \log_B(n/q) \rceil = O(n/q)$. All credits, lookup costs, and padded carry costs are functions of public occupancy and digit counters.

FIFO and helpers. Keep a client FIFO of address/value records, with validity bits and public occupied-slot count. A public write normally appends a record. A private read makes one authority read, overriding its result with the newest matching valid FIFO value; a private write makes one authority write and invalidates all older matching FIFO records without freeing slots. Add one dummy read from the authority to every private request, so that it makes two own authority calls. Invalid FIFO slots are removed by dummy authority calls; valid slots are removed by writing their supplied value to the authority. Draining the FIFO writes from oldest to newest preserves the logical array represented by the authority and the remaining FIFO.

A private request immediately following another private request is a *natural helper*. After G consecutive nonhelpers, the next request is a *forced helper*. Additionally, every public read is also a helper. A helper completes any due carry, drains the FIFO, executing carries whenever the authority buffer fills, serves its own request, and resets the helper counter. A public write acting as a forced helper writes directly to the authority. Non-helper private requests defer all carries. There are at most G FIFO slots and at most $2G$ authority calls whose carries are deferred between helpers. Public reads are served from the authority after draining the FIFO.

With FIFO occupancy F, take $\Upsilon = \Psi + eF$. Removing a slot releases e credits, covering the amortized cost of the authority call used to remove it. A private request, including helper work, has probes plus $\Delta\Upsilon$ at most $2e$; a public request—whether a regular write, forced helper write, or read—has $\Delta\Upsilon \le e$. The bound $\Upsilon = O(n)$ follows from $qe = O(n)$. A full public reset exports the authority canonically, applies the FIFO locally during a fixed scan, and rebuilds the authority, clearing the FIFO and potential; Corollary B.7 makes this cost $O(n)$ using $O(q)$ client words and $O(n)$ server words. Apply Lemma E.21 with $\alpha = 2e$, $\beta = e$, a padded integer e, and $V_* = O(n)$,

choosing

$$h = \left\lceil \frac{16e}{p} \ln(2 + n/p) \right\rceil, \qquad \epsilon_C = (2 + n/p)^{-4}.$$

The bank uses $O(\log_2 n)$ bits, gives $P = O(BL/p)$ pointwise, and adds expected $O(n(1 + \epsilon_C \mathbb{E}\tau))$ reset work through any stopping prefix with finite expected value. A reset also clears the helper counter, so it can only delay the next forced helper.

Localized-charge accounting. Selected private requests follow public writes, so, unless they are forced helpers, each pays only its two own lookups, at most $O(L)$ probes. At any position, being a forced helper requires that the preceding G positions contain no natural helper; considering disjoint adjacent mode pairs gives

$$\Pr\{\text{forced at a specified position}\} \le (1 - p^2)^{\lfloor G/2 \rfloor} \le n^{-c_G/3} =: \epsilon_G.$$

Every forced helper costs $O(n)$, by the bound on Υ and $\Delta\Upsilon$. If $\tau > N$ is the first private request after N, its tail satisfies $\Pr(\tau \ge N + j) \le (1 - p)^{j-1}$. Bounding a forced helper event in that tail by the smaller of this quantity and ϵ_G, and summing, bounds the expected number of forced helpers through τ by $N\epsilon_G + O(\epsilon_G(1 + \log_2(1/\epsilon_G))/p)$. Our memory condition implies $p^{-1} \le n^{1/2}$ for large n; choose c_G sufficiently large so this forced-work contribution, after normalization, is $o(pL)$. Non-helper public writes themselves make no probes; their selected cost comes only from forced helpers and resets. The selected-private count is $ap\mathbb{E}\tau$ and the number of bridging writes is $\sigma a \mathbb{E}\tau$. Since $np = \omega(1)$, $\sigma = \Theta(1)$ and $\mathbb{E}\tau = N + 1/p = \Theta(N)$ for $N \ge n$. The lookup contribution to A_w is $O(pL)$, and added resets contribute $O(n/N + n\epsilon_C) = O(n/N + p)$, since $n\epsilon_C \le p$. This proves the asserted bound.

Dummy calls, invalidation, and schedule counters never depend on private values or returned answers; the authority's adaptive simulation and the FIFO invariant prove the security and correctness claims. Lemma E.24 bounds all remaining counters and refreshes keys at full authority renewals. Together with Convention B.3, this preserves the resource and prefix bounds on every tape, and Lemma E.25 applies to the complete closing tail. □

Corollary E.23 (Matching high private budgets)**.** *Fix $\theta, \varepsilon > 0$ and $0 < p \le 1/2$. Let $q = \Theta(m)$ satisfy*

$$\max\{n^\theta, Cp^{-2} \log_2 n\} \le q \le n^{1-\varepsilon}, \qquad H = \log_2(2 + n/(m + 1)),$$

and suppose

$$Q \ge CH/p, \qquad N \ge \max\{4n, n/p\},$$

for a sufficiently large fixed C. Here N is a prescribed polynomial deterministic horizon, closed at its next private request. For every fixed secure implementation family in the stated model with $\Pi_N \le Q$ (in particular, with $P \le Q$), under the asymptotic quantifier conventions of Appendix G,

$$A_w = \Omega\left(p \max\left\{1, \frac{H}{\log_2(2 + pQ/H)}\right\}\right).$$

Under the counted computational resources of Theorem E.22, this bound is attained up to constant factors by a parameterized secure family satisfying $P \le Q$, using $O(n)$ server words. Initialization remains separate.

Proof. The lower bound is Corollary E.5: the memory condition gives $\rho = \Theta(p)$, and the horizon satisfies $N \ge \max\{4n, 1/p\}$. For the attaining construction, put $s = n/q$ and $h = \log_2 s = \Theta(H)$. The assumption $Q \ge CH/p$ ensures that $V = pQ$ meets the threshold in Lemma C.2. Apply that lemma with c_* large enough to cover the private-cost constant in Theorem E.22, obtaining an integer

growth factor B with $c_* BL \le pQ$ and $L = O(\max\{1, H/\log_2(2 + pQ/H)\})$. The one-level endpoint $B = \lceil s \rceil$ supplies the constant floor. Theorem E.22 gives $P \le Q$ and $A_w = O(pL + n/N)$. Since $N \ge n/p$ and $L \ge 1$, we have $n/N \le p \le pL$, so $A_w = O(pL)$. Every worst-case-input budget bounds the average-input private cost, so it supplies both upper bounds. □

E.9 Bounded state on unbounded closing tails

Lemma E.24 (Bounded counters and renewable table randomness)**.** *The calendar of Theorem E.14, the authority/FIFO implementation of Theorem E.22, and the prefix implementation of Theorem E.27 admit finite-state realizations with the stated asymptotic client and server space. They retain a constant number of PRF keys and one shared evaluation workspace. Their structural counters and table-domain identifiers have $O(\log_2 n)$ bits throughout the execution, including an arbitrarily long closing tail. Every request has a public polynomial probe bound. Their pointwise private budgets and caps are unchanged up to the already allowed constant factors, and adaptive security and correctness hold on every prescribed polynomial execution.*

Proof. Calendar. Use a clock modulo $M_{\mathrm{clk}} = 12\,\mathrm{lcm}(v_J, v_*)$, with the scales from Theorem E.14; it is a multiple of every release period, is at least $12v_*$, and is $O(n^2)$. Release phases use this clock, and live jobs store deadline countdowns of at most $2v_*$ instead of unbounded absolute times. For an interval ending at e, any parent reservation ends before $e + 3v_*$, and a bottom snapshot removes it from the query view before $e + 3v_*$. A bottom snapshot job can pin it for at most another $2v_*$, so all its tables, archives, and build workspace are retired before $e + 5v_*$. Its record timestamps are greater than $e - v_i \ge e - v_*$, so their ages remain below $6v_*$. Client-buffer records satisfy the same bound: a buffer spans $q \le v_*$ requests, publishes its summary and thus leaves the query view within another $2q$ requests, and can be pinned for at most another $2v_*$.

When building a bottom snapshot with cutoff b, assign timestamp b to each snapshot output record while writing it to the snapshot. This does not change the represented state: the snapshot is the array at b, and later updates retain their later timestamps. This snapshot publishes at $b + 2v_*$, is replaced at $b + 3v_*$, and can remain pinned for at most another $2v_*$; its timestamps therefore have age at most $5v_*$. Thus all live timestamps, including those in pinned sources and in-progress outputs, have age less than $6v_*$. Store timestamps modulo M_{clk} and compare their ages relative to that clock; the age is less than $M_{\mathrm{clk}}/2$, so chronological comparisons are exact. Use a separate dummy flag for missing records.

Start a fresh independent PRF key every v_* requests and keep the last six keys in a cyclic buffer. Every table, including a temporary table, is created no earlier than its interval end or snapshot cutoff and is retired within $5v_*$ of that time. A key can be assigned throughout its era of length v_*, so all tables using it are retired before its buffer slot is reused six eras later. Each new table or temporary table receives its era's key and a distinct local domain identifier; only polynomially many tables are created in one era, so the identifier counter has $O(\log_2 n)$ bits. The schedule, bounded job credits, buffers, and lookup costs are unchanged.

Authority with a FIFO. A bottom carry retires all old authority tables; its only remaining younger records are the bounded client-buffer suffix. During this carry use a fresh key for new tables, retaining the old key until its sources are retired. In the new bottom, retag every surviving record with version one and renumber the remaining buffer suffix from version two in append order. The same renewal occurs at a full public reset. Between renewals there are $O(n)$ authority appends, including the at most $2G$ additional authority appends made while carries are deferred, hence polynomially many table creations. The version and domain counters are bounded; level digits, buffer occupancy, and the helper counter already were in the original construction. The credit bank is bounded by Lemma E.21.

Prescribed-prefix implementation. Before N, counters are bounded by the prescribed polynomial N. Each public-tail cycle in Theorem E.27 contains at most $S \le n$ writes; its full public reset retires all old tables, refreshes the key, and restarts local versions and domain identifiers. At most two keys coexist during a reset. After the switch to full scans, no table-domain or version counter is needed. If the entire array fits in client memory, no table keys or version counters are needed, and thus the state bound in this case is immediate.

Resources and simulation. Reserve a constant fraction of the client allowance for all retained PRF keys and the one shared evaluator; the additional key storage changes only constants, and calls are sequential. Domain identifiers never repeat within a key era. On any polynomial execution, polynomially many independent keys are generated, and a hybrid replacing their PRFs one key at a time gives the same random-function simulation proof for adaptively chosen requests. The public clock/renewal rules and correct timestamp comparisons preserve the caller invariants. Malformed records and primitive failures use Convention B.3; control counters still obey the finite bounds. Calendar quotas, a bounded number of $O(n)$ carries or resets in one request, and the prefix construction's bounded batches give a polynomial per-request probe bound on every tape. □

Lemma E.25 (Full closing-tail expectations)**.** *Fix polynomially bounded $N, 1/p$ and a realization from Lemma E.24, with per-request probe bound M_*. Let $\tau > N$ be the first subsequent private request on the two-class workload. Then the total number of probes through τ has finite expectation. For every integer $K \ge N$,*

$$\Pr\{\tau > K\} = a^{K-N}, \qquad \mathbb{E}[(\tau - K)_+] = \frac{a^{K-N}}{p}, \qquad \mathbb{E}\left[\sum_{t=K+1}^{\tau} \text{probes}_t\right] \le \frac{M_*}{p} a^{K-N}.$$

In particular, taking $K = N + \lceil p^{-1} \ln^2(2+\kappa) \rceil$ makes both the probability and expected probe count beyond K negligible. The same holds after normalization by the expected number of bridging writes.

Proof. The IID mode process gives the geometric identities. Sum $M_* \Pr\{\tau \ge t\}$ for $t > K$ to obtain the probes bound, which holds even on primitive-failure paths. Both K and M_*/p are polynomial, whereas $a^{K-N} \le \exp[-\ln^2(2+\kappa)]$ is negligible. The exact denominator from Lemma E.3 satisfies $\mathbb{E}D = \sigma a(N + 1/p) \ge a/\rho \ge 1/2$, so normalization preserves negligibility. Correctness and adaptive simulation on the polynomial execution through K, together with $\Pr\{\tau > K\} = \text{negl}(\kappa)$, apply to the execution ending at τ as well. The cost proofs use the full expectations; their pointwise budgets, caps, and memory bounds hold throughout the tail by Lemma E.24. □

E.10 The client memory needed for inexpensive public writes

This subsection separates a correctness-only memory obstruction from the privacy tradeoffs above. We use the two-class workload $\mathcal{W}_{W,R}$ and the closed localized charge of Definition E.2: $a = 1 - p$, τ is the first private request strictly after position N, with N fixed in advance, and $A_w = \mathbb{E}C/\mathbb{E}D$. Here $N \ge 1$ is arbitrary; the long-horizon convention for the read frontier does not apply.

Theorem E.26 (Finite-horizon memory obstruction)**.** *Fix a prescribed finite horizon $N \ge 1$, and let A_w be the localized charge on the execution ending at the first private request strictly after N (Definition E.2). Let $0 < p \le 1/2$, let public writes have independent uniform addresses and independent uniform w-bit values, and suppose $w \ge C \log_2 n$. For every $O(w)$ physical address size, there exists a sufficiently small constant $Q_{w,0} > 0$ such that for every implementation that is correct with probability at least $2/3$ on every fixed full-API input, uses at most mw client bits at every*

instant, and satisfies $A_w \le Q_w \le Q_{w,0}$,

$$m + 1 = \Omega\Big(\min\Big\{n, p^{-2}\log_2(1/Q_w),\ N + p^{-1}\log_2(1/Q_w)\Big\}\Big). \tag{48}$$

The conclusion already follows if only write probes are counted in A_w*. It needs no privacy assumption, no bound on private work, and no polynomial gap between* m *and* n*. Correctness is needed on prefixes of the closed execution extended by one private read. At* $A_w = 0$*,* $m = \Omega(n)$*: storing the entire logical array on the client attains zero ongoing probes with* $O(n)$ *client words after initialization.*

Proof. Put $m_0 = m + 1$, $\rho = p + a/n$, and $\sigma = p/\rho$. Write $P_j^{\rightarrow}$ for the number of physical writes at request j. The exact thinning identity (35), applied separately to probe directions, gives

$$\mathbb{E}C^{\rightarrow} = \mathbb{E}\sum_{j\le\tau}\omega_j P_j^{\rightarrow}, \qquad \mathbb{E}D = \sigma a(N + 1/p), \tag{49}$$

where $\omega_j = \sigma$ at public writes, $\omega_j = 1$ at private requests immediately preceded by a public write, and $\omega_j = 0$ at the other private requests.

An interval write bound. There are absolute constants $C_0, c > 0$ such that an interval of length Λ, with $C_0 m_0 \le \Lambda \le n/128$ and at least $\Lambda/2$ public writes, has expected physical-write cost at least $c\Lambda$, conditional on its mode pattern. To see this, first expose its addresses. With probability at least a fixed positive constant they leave at least $\Lambda/8$ distinct surviving written addresses, since the expected number of collisions among at most Λ independent uniform addresses is at most $\Lambda^2/(2n)$. The final value at each surviving address is an independent uniform word, also independent of the pre-interval state. Give a decoder the pre-interval state, all addresses, overwritten payloads, and the independent random bits used by the implementation at each request, but not these surviving words.

If the interval uses at most $\delta\Lambda$ physical writes, an encoder can transmit the post-interval client state and a padded list of every cell changed during the interval, with its address and final contents; physical addresses have $O(w)$ bits. For sufficiently small fixed δ and large C_0, this fixed-length message occupies at most one sixteenth of the total surviving-word length. From it the decoder reconstructs the complete post-interval state and simulates an appended private read of an independently selected uniform surviving address. If the interval uses more than $\delta\Lambda$ physical writes, the encoder sends an abort flag and padding. As in Convention A.4(d), truncate the decoder's local simulation at a finite, possibly inefficient, cutoff losing at most $1/64$ success probability. Lemma A.2 then implies that the abort event has constant probability, since genuine appended reads are correct with probability at least $2/3$. Hence the expected number of physical writes is $\Omega(\Lambda)$.

Call an interval *admissible* if it starts with a public write and contains no consecutive private pair. Every private request in such an interval immediately follows a public write, so every request in it has weight $\omega_j \ge \sigma$. Hence, for any family $\mathcal{I}$ of disjoint admissible intervals, the weighted sum in (49) is, pointwise, at least σ times the number of physical writes made inside the intervals of $\mathcal{I}$. An admissible interval of length Λ has at least $\Lambda/2$ public writes, so the $\Omega(\Lambda)$ interval write bound above applies to it whenever $C_0 m_0 \le \Lambda \le n/128$: its expected physical-write cost is at least $c\Lambda$.

One point needs care here. Whether an interval is admissible depends on modes that come after the interval starts, so $\mathcal{I}$ is not fixed in advance. This does no harm, because $\mathcal{I}$ is determined by the modes alone, and the interval write bound holds conditionally on the modes. Fix the whole mode sequence: this fixes $\mathcal{I}$, and since addresses and payloads are independent of the modes, the bound still gives expected physical-write cost at least $c\Lambda$ for each interval of $\mathcal{I}$. Averaging over the modes afterwards gives

$$\mathbb{E}C^{\rightarrow} \ \ge\ \sigma c\Lambda\,\mathbb{E}|\mathcal{I}|$$

for every such family of length-Λ intervals.

Prefix intervals and the closing tail. Let s_j be the probability that j independent modes contain no consecutive private pair, so $s_0 = s_1 = 1$ and $s_j = as_{j-1} + aps_{j-2}$ for $j \geq 2$. A length-Λ block begins with a public write and has no private pair with probability $as_{\Lambda-1}$; in such a case, it has at least $\Lambda/2$ public writes. Partition the first N requests into $\lfloor N/\Lambda \rfloor$ consecutive blocks of Λ requests, ignoring any remainder. Separately, partition the potential closing tail after N into length-J blocks; its jth block lies before τ and is entirely public with probability a^{jJ}. For any $C_0 m_0 \leq \Lambda, J \leq n/128$, the interval write bound and (49) yield

$$A_w^{\rightarrow} \geq \frac{c}{N + 1/p} \left(\left\lfloor \frac{N}{\Lambda} \right\rfloor \Lambda s_{\Lambda-1} + \frac{Ja^{J-1}}{1 - a^J} \right). \tag{50}$$

One may retain just $\max\{1, \lceil 1/(pJ) \rceil\}$ tail blocks, which contribute a constant fraction of the displayed geometric sum; therefore, this proof only requires correctness through $N + O(J + 1/p)$ positions and their one-query extensions.

Inverting the bound. Set

$$\lambda = \frac{a + \sqrt{a^2 + 4ap}}{2}, \qquad k_1 = \log_2(1/a), \qquad k_2 = \log_2(1/\lambda), \qquad \ell = \log_2(1/Q_w).$$

The recurrence gives $s_j \geq \lambda^j$ by induction. Since $1 - \lambda = p^2/(p + \lambda)$, for $p \leq 1/2$ we have $k_1 = \Theta(p)$ and $k_2 = \Theta(p^2)$. If $m_0 \geq n/(256C_0)$, (48) is immediate. Otherwise take $\Lambda = J = \lceil C_0 m_0 \rceil \leq n/128$, adjusting constants for bounded n.

Write $B = \ell/k_1$ and $Q = \ell/k_2$. If $N \leq B$, then $1 - a^J \leq pJ$ and $pB = O(\ell)$, so the tail term in (50) gives $Q_w \geq c\, 2^{-C_0 k_1 m_0}/(1 + \ell)$. Taking logarithms, and choosing $Q_{w,0}$ sufficiently small to absorb $\log_2(1+\ell) + O(1)$, gives $m_0 = \Omega(B)$, as required since $\min\{Q, N + B\} = \Theta(B)$. If $N > B$ but $N < 2\Lambda$, then $m_0 = \Omega(N)$ and $\min\{Q, N + B\} \leq 2N$. Finally, if $N > B$ and $N \geq 2\Lambda$, the full prefix blocks cover at least $N/2$ positions and $N + 1/p = O(N)$; the prefix term gives $Q_w \geq c2^{-Ck_2 m_0}$, hence $m_0 = \Omega(Q)$. These cases establish (48).

For the $A_w = 0$ case, notice that all-public closing-tail segments of every fixed finite length have positive probability. If $m = o(n)$, one such segment of length $\Theta(m + 1) \leq n/128$ has positive expected physical-write cost by the interval write bound, contradicting (49); thus $m = \Omega(n)$. This can be realized by storing the entire array in client storage. □

Theorem E.27 (A memory-efficient prescribed-prefix construction)**.** *Fix $\eta > 0$, let $0 < p \leq n^{-\eta}$, and let $N \geq 1$ be a prescribed polynomial horizon. For sufficiently large C and small fixed $Q_{w,0} > 0$, let*

$$Cp \leq Q_w \leq Q_{w,0}, \qquad B_{Q_w} = p^{-1} \log_2(1/Q_w).$$

Under the computational assumptions of Section 2.4, with the retained PRF keys and shared evaluation workspace fitting in the chosen client allowance, there is a full-API implementation with

$$m = O(\min\{n, N + B_{Q_w}\}), \qquad A_w \leq Q_w, \tag{51}$$

$O(n)$ server words, and separately charged $O_\eta(n)$ initialization. Every private request on every input consisting of public writes and private operations through the first private request after N costs $O(1)$ probes. When $N + B_{Q_w} \leq p^{-2} \log_2(1/Q_w)$, its memory is optimal up to constants by Theorem E.26.

Proof. Let $B = \lceil C_B B_{Q_w} \rceil$, and fix a sufficiently small constant $c > 0$. If $N + B > cn$, store the entire array in client storage. Otherwise, $N + B \leq cn$. Since $B \geq n^\eta$, a fixed fraction of the $O(N + B)$

client-memory budget is polynomial workspace. The sorter and fresh shuffle used in Corollary B.7 build a new consuming table of capacity $s \le O(n)$ in $O_\eta(s)$ probes, including sorting by address and keeping the newest timestamped record. All table stashes fit within the same allowance; only a constant number of tables are live.

Initialize a consuming table containing the initial array. During the first N requests, keep a client dictionary of the newest value at every touched address. A public write updates the dictionary locally. A private request checks the dictionary, performs one padded initial-table lookup, real only on a dictionary miss, and then caches its returned or supplied value. Each real initial-table query is therefore distinct, including under adaptive request selection. The dictionary has at most N entries and each private request costs $O(1)$ probes.

After position N, freeze this dictionary and collect subsequent public writes in a client buffer of B timestamped records. At each full batch, retire the current tail table, if any, and merge its records with the buffer by an oblivious sort, keeping the newest version at each address, and build a fresh, independently shuffled tail table padded to capacity jB after the jth batch; such a rebuild costs $O(jB)$. No tail table has been queried before its retirement. At the closing private request, search the current buffer locally, issue one padded tail-table lookup if a tail table exists, and consult the prefix dictionary and initial table, with dummy lookups wherever a newer value is already known. Every initial-table real query is still nonrecurrent, and the final tail table receives at most one real query, so the closing request also has $O(1)$ cost. Retain one extra client record for its answer or supplied write value.

To bound space on arbitrarily long closing tails, restart after $S = B\lfloor n/B \rfloor \in [n/2, n]$ public tail writes. Retire the initial and tail tables and combine their surviving records with the prefix dictionary and current buffer. A fixed scan, oblivious sort, and latest-version filter recover the current array: every value consumed from the initial table is represented in the prefix dictionary, and newer tail values supersede it. Build one fresh initial table, clear the other structures, and restart the public-tail counter. This public reset costs $O(n)$ probes; repeated resets change neither the constant private cost through the closing request nor the $O(n)$ server-space bound. At the request after closure, or immediately upon a public read, perform a full oblivious reset and continue with a fixed full-array scan per request. These switches are determined by modes and public counters. The continuation supports the full API and uses the same memory, but it is outside the stated two-class prefix cost guarantee.

The consuming-table simulators, fresh independent builds, and fixed scan/sort resets give adaptive trace privacy. A failed primitive uses Convention B.3, and errors remain negligible on prescribed polynomial executions.

It remains to account for the charged closing tail. Its public length $T = \tau - N - 1$ satisfies $\Pr(T \ge t) = a^t$. Put $z = a^B$. Since $pB \ge C_B \log_2(1/Q_w)$ is a large constant, $z \le 1/2$ and $a^S \le 1/2$. Summing batch costs over all possible tail cycles gives

$$\mathbb{E}[\text{tail rebuild probes}] \le \frac{CB}{1 - a^S} \sum_{j \ge 1} j z^j = O(Be^{-pB}),$$

and the public cycle resets contribute $O(ne^{-pn/2})$. All public-write work in the first N positions is zero. Every selected private request costs at most a fixed constant, so the expected request counts through τ and thinning identity (35) imply

$$A_w \le C' \left(p + \frac{pBe^{-pB} + pne^{-pn/2}}{1 + pN} \right). \tag{52}$$

Here $p/\sigma = p + a/n = O(p)$, because the case $N + B \le cn$ has $n \ge 4B$ for sufficiently small c and pB is bounded below by a positive constant. Increase C_B so that both exponential terms are

at most $Q_w/2$ after multiplication by C', and increase the constant C so that $C'p \le Q_w/2$. This proves (51). Realize the public-tail cycles with the renewed keys and bounded local versions of Lemma E.24. In the case $N + B \le cn$, the bound $B \le cn$ also implies $1/p = O(n)$, so Lemma E.25 applies: the displayed sums account for the entire tail, and the polynomial cutoff is needed only for the computational experiment; in particular, it never replaces a closing private lookup by a full scan. Full client storage handles the remaining cases exactly. □

Corollary E.28 (The complete finite-horizon memory curve in a matched regime)**.** *Fix $0 < \eta < 1/2-\delta$ and let $n^{-1/2+\delta} \le p \le n^{-\eta}$. Under the counted computational resources of Corollary E.16, for $Cp \log_2 n \le Q_w \le Q_{w,0}$ and every prescribed polynomial $N \ge 1$, the least client memory permitting $A_w \le Q_w$ is*

$$\Theta\Big(\min\Big\{n, p^{-2}\log_2(1/Q_w),\ N + p^{-1}\log_2(1/Q_w)\Big\}\Big). \tag{53}$$

The PRF key and evaluation workspace must fit in this memory allowance. The attaining implementations use $O(n)$ server words and $O(n)$ separate initialization. In the prefix branch they have $O(1)$ private cost through closure; in the long-horizon branch the calendar simultaneously attains the private budget and cap of Corollary E.16.

Proof. Theorem E.26 proves the lower bound without computational assumptions. Choose the smallest of full client storage, Theorem E.27, and the calendar of Corollary E.16, whose respective memory bounds are $O(n)$, $O(N + p^{-1}\log_2(1/Q_w))$, and $O(p^{-2}\log_2(1/Q_w))$. The common target interval is contained in each construction's range. Since the memory bound is $\omega(1)$, so the additive term $+1$ in the lower bound is absorbed. □

F General workloads: cost bounds and construction

This appendix proves the three theorems of Section 3.3 and gives the matching construction. Theorem 3.5 splits by the rate $u = a + p$ of potential writes (Section 4.3): Lemma F.1 (a private terminal) covers $u \ge 1/4$ and Lemma F.2 (a public terminal) covers the rest; both are instances of Lemma A.6. Theorem 3.6 combines the same two lemmas. The public write bound (Theorem 3.7) follows by inverting the private-terminal tradeoff and using the total-cost bound. The matching construction (Theorem F.4) combines the eager calendar of Theorem E.14 and Lemma E.13 with the public index and cache of Appendix B.3.

F.1 Average cost bounds on deterministic logical horizons

We prove Theorem 3.5, Theorem 3.6, and Theorem 3.7. Throughout this subsection put

$$u = a + p = 1 - r, \qquad x = \frac{n}{m+1}, \qquad H = \log_2(2 + x), \qquad D = \log_2(2 + 1/p).$$

The rate u counts public writes and private operations, including private reads that can be replaced by writes under hidden-kind privacy; actual writes have rate $a + p\xi \le u$. All expected costs below use the same prefix of fixed length N logical requests. Unlike the two-class write proofs, the comparisons here change private kinds and therefore require hidden private kinds.

Lemma F.1 (Private terminals in a three-class workload)**.** *Under the assumptions of Theorem 3.5,*

$$T_N/u = \Omega(1), \qquad \Pi_N = \Omega(1), \qquad \Phi(T_N/u, \Pi_N) = \Omega(H). \tag{54}$$

Consequently

$$T_N = \Omega\Big(u \max\Big\{1, \frac{H}{\log_2(2 + u/p)}\Big\}\Big). \tag{55}$$

Proof. Choose an endpoint uniformly from $\{\lceil N/2\rceil, \dots, N-1\}$ and append a private reference request with its independent workload kind, address, and, when needed, write value. By online execution, the terminal has the reference request's cost distribution conditioned on being private. Averaging over the chosen positions and using the definition of Π_N gives a loose bound $Y \le 4\Pi_N$, where Y is the terminal's expected cost. Put

$$\omega = \left\lfloor \frac{n}{1024u} \right\rfloor, \qquad \ell_0 = \left\lceil \frac{C(m+1)}{u} \right\rceil.$$

These windows fit before all endpoints, since $u \ge p$ and $N \ge 4n/p$, and their logarithmic range is $\log_2(2 + \omega/\ell_0) = \Theta(H)$.

For an epoch I of length $\ell \ge \ell_0$, condition on it having at least $u\ell/2$ *active positions* (public writes or private operations); this has probability $g \ge 63/64$. Select its first $k = \lfloor u\ell/4 \rfloor$ active positions. At selected public writes keep their fresh uniform values; replace each selected private request by a fresh uniform write at its original uniform address. The k selected source words are independent and uniform. Leave every other prefix request unchanged. After this prefix has been specified, choose independent uniform $J \in [k]$ and make the altered terminal a private read of the Jth selected address.

The expected number of active positions in the entire window, conditional on I containing at least $u\ell/2$ active positions, is at most $u\omega/g$, and all addresses are still independent uniform variables conditional on the modes and J. Thus the chance that a later write overwrites the selected source word is at most $u\omega/(gn) < 1/512$; counting every active position as a write only enlarges this bound. The altered terminal therefore returns that word with probability at least $2/3 - 1/512$.

Let E_I count physical-read dependency edges from I to younger requests, and D_I such edges into the terminal. Give the entropy decoder request modes, addresses, all non-source values, the independent reference private values, pre-epoch state, and implementation tapes as side information. The message sent by the encoder before J consists of the post-epoch client state and $\lfloor \eta u\ell \rfloor$ padded repair records, or an abort flag and padding; its length is at most $kw/16$ for small η and large C. The same induction over simulated probes that shows the decoder's correct reconstruction and the same finite decoder timeout as in Appendix A.2 apply here. Correctness, Lemma A.2, and transfer of the observable event $\{E_I > \lfloor \eta u\ell \rfloor \vee D_I > 0\}$ from the altered to the reference workload give

$$\mathbb{E}D_I + \frac{\mathbb{E}E_I}{\eta u\ell} \ge c, \tag{56}$$

and removing the active-position-count conditioning costs only a constant factor.

Lemma A.6 applies with density $d = u$, the above window and shortest epoch, a conditioning event with probability $g = 1$ (the active-position-count conditioning having been removed), the above private reference terminal, and $X = T_N/u$; its part (i) gives $Y = \Omega(1)$. Its part (iii) needs the floor $X = \Omega(1)$, that is, $T_N = \Omega(u)$. To supply it, apply the same altered experiment to disjoint length-ω intervals covering a constant fraction of the first $N/2$ requests, with an appended private query immediately after each interval. A message containing the post-interval client state and the final address/value pairs of all physically written cells reconstructs its complete final state. If the interval writes fewer than $\eta u\omega$ cells, a padded version of this message has at most $kw/16$ bits and recovers the selected source word, apart from the previously bounded probabilities of overwriting the selected source word and exceeding the decoder's simulation cap. Lemma A.2 forces a constant-probability event of $\Omega(u\omega)$ physical writes in the altered execution; transfer that bounded past-trace event to the reference and sum the disjoint intervals to obtain $T_N = \Omega(u)$. This needs hidden-kind privacy when fresh words were introduced by replacing reference private reads with writes.

Lemma A.6(iii) now gives $\Phi(T_N/u, Y) = \Omega(H)$. Since $Y \le 4\Pi_N$, monotonicity and constant-factor homogeneity give (54). Finally, $p\Pi_N \le T_N$ and $\Phi(s,t) \le Cs\log_2(2+t/s)$ imply

$$H \le C\frac{T_N}{u}\log_2\!\left(2+\frac{u\Pi_N}{T_N}\right) \le C\frac{T_N}{u}\log_2(2+u/p),$$

which together with the floor proves (55). □

Lemma F.2 (Public terminals in a three-class workload)**.** *Suppose also $r > 0$. Then $R_N = \Omega(1)$. If $xp/u \ge C_0$ for a sufficiently large constant C_0, put $h_{\mathrm{pub}} = \log_2(2+xp/u)$. Then*

$$R_N \log_2\!\left(2+\frac{T_N}{pR_N}\right) = \Omega(h_{\mathrm{pub}}). \tag{57}$$

Proof. A constant lower bound for public reads. Choose a uniform endpoint e in the second half of the prefix and let S be the addresses touched by active positions through e. Since $ue \ge 2n$, the expected number of untouched addresses is at most ne^{-2}; hence $|S| \ge n/2$ with probability at least $1-2e^{-2}$. Condition on this event. Keep every public request and replace every private request in the prefix by a fresh independent write at its original address. At each address in S, its final value is the word at its last active position; these values are independent uniform words conditional on modes and addresses. After the prefix, choose a uniform address from S and issue the same public read of that address in the altered and reference inputs. These inputs are comparable. The reference query has conditional density at most two relative to a uniform address; conditioning the prefix on the touched-address count and restricting the endpoint to the second half of the prefix also cost only constant factors. Its expected cost is therefore $O(R_N)$.

Send only the altered execution's preterminal client state, of at most mw bits, before revealing the queried coordinate. The decoder knows all request fields other than the last-active source words, together with the independent tapes. It simulates the public terminal until its first physical read, aborting there; overwrites reveal no old contents and can be simulated without knowing the server state. A finite local timeout again loses at most $1/64$. On the event of no physical read the simulation is exact, and Lemma A.2 bounds recovery of a uniform coordinate by $m/|S| + 1/w = o(1)$. Correctness thus forces a constant-probability physical-read event, which transfers to the reference. This proves $R_N = \Omega(1)$.

The logarithmic term. Use the fixed-public-query proof with

$$\omega = \left\lfloor \frac{n}{128u} \right\rfloor, \qquad \ell_0 = \left\lceil \frac{C(m+1)}{p} \right\rceil.$$

The safe set excludes every address written by a reference public or private request in this window. The expected number of excluded addresses is at most $u\omega \le n/128$, so the event of safe-set size at least $15n/16$ still has probability at least $7/8$. An epoch supplies $k = \lfloor p\ell/4 \rfloor$ replaceable private positions with constant conditional probability. Replace them by fresh writes at distinct safe addresses and use the same terminal public query in the two executions. All public writes, their values, and all public reads remain unchanged; hidden kinds allow both private reference kinds to supply source positions. The inserted source words survive until the terminal because their addresses are distinct and all unchanged writes in the window avoid the safe set. The decoder has the same side information as in Lemma 4.1, including all unchanged request fields and public-write values. The same induction over simulated probes therefore applies: supplied cell records correct reads of cells changed during the skipped epoch, and subsequent writes can be simulated from the known request fields, regardless of whether the requests are public or private.

For xp/u sufficiently large, $\omega \geq 2\ell_0$ and the available logarithmic range is $h_* = \log_2(2+\omega/\ell_0) = \Theta(\log_2(2+xp/u))$. Lemma A.6(ii) applies with density $d = p$ and the safe-set-size event as G_0; its per-edge sum charges *all* younger probes, including public write probes, so $X = T_N/p$, and the reference terminal has cost $Y \leq 4R_N$. The increasing-function substitution of Appendix A.5 proves (57). □

Computational qualifications. Every experiment above has at most N logical requests, and the conditioning events requiring enough eligible positions in an epoch and a sufficiently large safe set each have probability bounded below by a positive constant. The forced terminal has expected cost $O(\Pi_N)$ when private and $O(R_N)$ when a public read. The reference prefix retains its original expected-probe bound up to constant conditioning factors, since forcing the terminal's class does not change the prefix distribution. For private terminals, $\Pi_N \leq T_N/p$ and polynomially bounded $1/p$ therefore give a polynomial expected cost for the full reference execution. For the public read terminal, if R_N exceeds a sufficiently large constant multiple of H, the public-read lower bounds hold immediately. In the remaining case, $R_N = O(H)$, so the forced public-read terminal has polynomially bounded expected probe count, as required by the transfer lemma. This justifies public terminals even for negligible $r > 0$. The remaining hypotheses of Lemma A.3 follow from Convention A.4: $1/p$ is polynomial because $N \geq 4n/p$ is, all samplers use efficiently sampleable request fields and polynomially bounded arrays of addresses, and nonuniform advice selects the epoch scales.

Proof of Theorem 3.6. Lemma F.1 gives $T_N \geq cu$, while Lemma F.2 gives $R_N \geq c'$. The function $f(y,t) = y\log_2(2+t/(py))$ is increasing in both positive coordinates, so these floors imply $f(R_N, T_N) = \Omega(\log_2(2+u/p))$. If xp/u exceeds the fixed threshold of Lemma F.2, its logarithmic bound also gives $f(R_N, T_N) = \Omega(\log_2(2+xp/u))$; below that threshold this second logarithm is bounded, so the bound $f(R_N, T_N) = \Omega(\log_2(2+u/p))$ already suffices. Finally, $H = \log_2(2+x) \leq \log_2(2+u/p) + \log_2(2+xp/u)$, which proves the tradeoff bound with uniform constants. □

Corollary F.3 (Private and public-write budgets)**.** *Under Theorem 3.6's assumptions, if $\Pi_N \leq Q$ and $W_N \leq V$, then*

$$R_N = \Omega\left(\max\left\{1, \frac{H}{\log_2(2+r/p+Q/H+aV/(pH))}\right\}\right).$$

An absent public-write class contributes zero. A worst-case-input private budget Q implies the same bound.

Proof. Substitute $T_N = p\Pi_N + aW_N + rR_N$ into (8), then apply Lemma C.1 with $A = 2 + r/p$ and $D = Q + aV/p$, retaining the constant floor. □

Proof of Theorem 3.5. If $u \geq 1/4$, (55) gives $T_N = \Omega(1)$ and

$$T_N \geq c\frac{uH}{\log_2(2+u/p)} \geq \frac{c}{4}\frac{H}{D}.$$

This includes $p > 1/2$ and the endpoint $r = 0$.

Otherwise $r = 1 - u > 3/4$, so Lemma F.2 gives $T_N \geq rR_N = \Omega(1)$. If $D \geq H/4$, the desired right-hand side is at most a constant and this floor suffices. If $D < H/4$, then $p > 2^{-H/4}$ and

$$\frac{xp}{u} \geq xp > \frac{x}{(2+x)^{1/4}}, \qquad h_{\mathrm{pub}} \geq \frac{3}{4}H - O(1) = \Omega(H).$$

For sufficiently large n, xp/u exceeds the threshold in Lemma F.2, so (57) gives

$$R_N \log_2\left(2 + \frac{T_N}{pR_N}\right) = \Omega(h_{\mathrm{pub}}) = \Omega(H).$$

The function $y \log_2(2 + z/y)$ is increasing in $y > 0$; using $R_N \le T_N/r$ in (57) therefore gives $H \le C(T_N/r)\log_2(2 + r/p) \le C' T_N D$. Together with the constant floor, this is (7). The cases exhaust every split $p + r + a = 1$. □

Proof of Theorem 3.7. Lemma F.1 gives the floors and the tradeoff in (9). For the budget form, put $X = T_N/u$, so $\Phi(X, \Pi_N) \ge c_1 H$. Since $\Phi(X, y) \le CX \log_2(2 + y/X)$ for positive X, y, the bound $\Pi_N \le Q$ and Lemma C.1 with $A = 2$, $D = Q$, and $S = H$ give

$$T_N = uX \ge c_2 \frac{uH}{\log_2(2 + Q/H)}.$$

Together with $T_N \ge c_3 F_p$ from Theorem 3.5, this gives $T_N \ge c \max\{F_p, uH/\log_2(2 + Q/H)\}$ for a uniform $c > 0$. Finally,

$$aW_N = T_N - p\Pi_N - rR_N \ge T_N - pQ - rV$$

and $W_N \ge 0$ give (10). A worst-case-input private budget implies that an equal average-input budget is also satisfied. □

F.2 Matching total cost for general workloads

For the total-cost construction, we replace the read-frontier construction's authority hierarchy with the write calendar of Theorem E.14, while retaining its public index and cache. The calendar answers private requests and accepts public writes by appending their new values without searching for earlier versions; the public index and cache serve public reads. Each public write also updates the public cache, keeping public answers consistent with the calendar without the full rebuild of Procedure 3.

Theorem F.4 (A three-class construction and optimal total cost)**.** *Fix $0 < \theta < 1 - \varepsilon < 1$ and $n^\theta \le m \le n^{1-\varepsilon}$. Assume the word-size and counted PRF-workspace conditions of Section 2.4. Put $q = \lfloor c_m m \rfloor$ for a sufficiently small fixed $c_m > 0$. For an integer $2 \le B \le \lfloor n/q \rfloor$, put*

$$L = \lfloor \log_B(n/q) \rfloor + 2, \qquad BL \log_2 n \le cq,$$

where $c > 0$ is a sufficiently small allocation constant. There is a full-API implementation with hidden-kind adaptive computational privacy, at most m client words, $O(n)$ server words, and $O(n)$ separately charged initialization. Each public request costs $O(L)$ probes. On every input prefix, the total private probes are at most CBL times the number of private requests, on every tape. Consequently, on any workload with IID request classes and private probability $0 < p \le 1$, and arbitrary adaptive choices of addresses and values,

$$R_N, W_N = O(L), \qquad \Pi_N \le P = O(BL), \qquad T_N = O\big(L(1 + pB)\big).$$

An absent public class has cost zero. Choosing

$$B = \min\Big\{\lfloor n/q \rfloor, \lfloor q^{1/4} \rfloor, \max\{2, \lfloor 1/p \rfloor\}\Big\} \tag{58}$$

gives $T_N = O(F_p)$, uniformly over the public read/write split. On the uniform-address workloads and deterministic horizons of Theorem 3.5, the optimal total cost is therefore $\Theta(F_p)$ in this resource regime.

Proof. Write calendar for private requests. Use the write calendar of Theorem E.14 with eager service as the authority. Its fixed release and publication times depend only on the logical-request count. A private request searches its current view, then appends its address and either the retrieved value or its supplied write value; both private kinds perform the same padded search. A public write appends its supplied record without searching. A public read occupies one dummy append slot and makes no authority lookup. Every request performs the eager maintenance quotas.

Lemma E.13 also permits dummy records: occupancy and deadlines count slots, while a summary retains the newest real record for each key and pads the rest. A private lookup is protected against recurrence by its subsequent real append until the queried table retires; public-write appends can only replace that protection by a newer record, and dummy appends do not remove it. The calendar's authority correctness, fixed deadlines, and immutable-archive rebuilding therefore remain valid. Its lookup cost is $O(BL)$ and its maintenance is $O(L)$ per logical request, on every tape.

Public index and cache. Independently maintain the hierarchical consuming-table index of Section B.3, with growth factor B, immediate carries, and appends only on private requests. Each private request appends the same logical address and resulting value as to the authority, using independent representation randomness; cache records never enter index merges. The index is initially empty; the public cache $C[x]$ initially contains the input value of x, version zero, and a sentinel public-touch time.

A global version counter increases at every private request and public write. A private index append receives that fresh version. A public write stores its supplied value and fresh version directly in $C[x]$, and advances $\mathrm{last}(x)$ to the completion time of this public write; it neither searches nor appends to the index. Public reads use Procedure 2's clean-index query phase: read $C[x]$, inspect the local index buffer, query each active index table S with x precisely when $\mathrm{last}(x) < \mathrm{creation}(S)$, and use a dummy otherwise; take the largest version among all candidates, cache it, and advance $\mathrm{last}(x)$ to this read's completion. All active tables are queried, including after a hit. A common event clock strictly orders public completions and index table creations. Private index carries finish immediately before their request returns; there is no dirty index or public reset.

Correctness of public answers. Every current value is represented either by the cache or by a live index record: a private request creates a fresh record, and a public write creates a newer cache record. Index merges remove a record only in favor of a newer record, and public consumption removes it only after the cache has acquired a version at least as large; cached versions never decrease. These cache-domination invariants from Section B.3 also hold when a public write directly updates the cache, because its fresh version is newer than every existing record for that address.

A table receives a dummy query for x only if a public read or write to x completed after its creation. That access already established cache domination over the table's records for x. Tables created later receive real queries, and any stale records they return lose the version comparison with the cache. The calendar handles private answers, so correctness holds for the full API.

Privacy and counted resources. An index table receives at most one real public query for a given key: the first such query advances the public-touch time beyond its creation, and later writes only increase that time. Internal merges use the retiring construction of Lemma B.4, and authority nonrecurrence was proved above. Both components therefore satisfy the consuming-table interfaces, and their independent builds preserve the joint shuffle invariant.

Publication times, padded lengths, index carries, and real-versus-dummy public-query choices depend only on disclosed modes, public addresses, and counters. Cache probes use only disclosed addresses, and the returned values and their versions do not change the probe schedule. The reactive simulation argument of Theorem B.9, together with Lemma E.11, thus applies to the combined implementation, including adaptive logical requests and hidden private kinds. Convention B.3 preserves every stated probe and storage bound and contributes only negligible error on each

prescribed polynomial execution.

The authority has $O(BL)$ live lookup tables and counted stash and job state $O(BL \log_2 n)$; its buffers and total build workspace use $O(q)$ words. The index needs $O(q)$ buffer/workspace words and $O(L \log_2 n)$ retained table state. Index carries and calendar maintenance execute sequentially within each request, so index carries do not affect the calendar's deadlines. Choose c_m, c to fit the above memory allocations and the shared PRF evaluator within m words. Since q is polynomial in n, every live authority build can retain polynomial workspace as in Theorem E.14. Independent table PRF domains use a constant number of keys and $O(\log_2 n)$-bit counters on the prescribed polynomial horizon. The calendar, index, and public cache together use $O(n)$ server words. Initialize the calendar's bottom snapshot using Lemma E.11 and copy the input array into the public cache, at total cost $O(n)$; the index starts empty.

Cost and growth factor. Every public request pays $O(L)$ authority maintenance; a public read also pays $O(L)$ index lookups, and either public kind makes only constantly many public-cache probes. Private authority costs are $O(BL)$ per request. If a prefix contains I private requests, the index has exactly I appends, and (24) bounds all its private carry work by $CBLI$. This proves the all-prefix private budget and, after averaging $\mathbb{E}I = pN$, the stated average-cost bounds.

For sufficiently large n, the growth factor B in (58) is at least two and satisfies $BL \log_2 n = o(q)$ and $pB \le 2$. If $B = \max\{2, \lfloor 1/p \rfloor\}$, then $L = O(1 + H/\log_2(2 + 1/p)) = O(F_p)$; $B = \lfloor n/q \rfloor$ gives $L = O(1)$, and $B = \lfloor q^{1/4} \rfloor$ gives $L = O_\theta(1)$, because $q = \Omega(n^\theta)$. Thus $L = O(F_p)$ in every case. Combining the upper bound with Theorem 3.5 proves the claimed total-cost optimum. □

G Computational parameters and encrypted storage

This appendix collects the computational conventions behind Sections 2.2 and 2.4: the definition of computational trace privacy, the quantifiers behind the asymptotic statements, the PRF instantiations, and a wrapper that additionally encrypts cell contents.

Computational trace privacy. Statistical trace privacy on horizon $\bar{N}$ requires that the traces of any two comparable inputs of length at most $\bar{N}$ have total-variation distance at most $1/16$, and unrestricted statistical privacy requires this for all finite lengths. For statistical lower bounds on a deterministic horizon of N logical requests, privacy on horizon $N + 2$ suffices: every comparison in their proofs (reference prefix, altered prefix, and terminal block) ends by position $N + 1$. The closing tails of localized charges enter only the cost accounting, never a privacy comparison. Lemma E.8 instead fixes a horizon of N macros, whose public-read runs have unbounded support when $r > 0$; its statistical lower bound assumes unrestricted statistical privacy for the expanded logical inputs. For computational privacy, fix a security parameter κ, and let $\mathcal{A}_\kappa$ denote the randomized ORAM implementation at that security parameter. For every nonuniform polynomial-time sampler that produces comparable inputs (u_0, u_1) and polynomial-length auxiliary information z independently of the implementation's coins, and every polynomial probe cutoff $b(\kappa)$, we require

$$(z, \mathsf{Tr}_{\le b}(\mathcal{A}_\kappa(u_0))) \approx_c (z, \mathsf{Tr}_{\le b}(\mathcal{A}_\kappa(u_1))),$$

where $\mathsf{Tr}_{\le b}$ is the trace truncated after b probes, with a marked overflow if the execution exceeds the cutoff, and $\approx_c$ denotes indistinguishability against nonuniform polynomial-time distinguishers, which, like the samplers, may use polynomial-length advice depending on κ. Public parameters may be part of z.

The lower bounds require only privacy for input sequences chosen before execution. The constructions satisfy the stronger *adaptive trace simulation* of Definition B.8: a polynomial-time

Client words m	Word bits and parameter scaling	Matching upper bounds
$n^\theta \le m \le n^{1-\varepsilon}$	$w = \Theta(\log_2 n)$, $n = \kappa^d$; $d > \max\{1, c/\theta\}$	Covered with logarithmic words
$\lceil C_0 \log_2^{12}(2+n) \rceil$	$w = \Theta(\kappa^c)$, $n = \kappa^d$; $d \ge 1$	Covered with larger words
$\log_2^{O(1)} n$	$w = \Theta(\log_2 n)$, $n = \kappa^d$; fixed d	Not covered: a conventional key exceeds the client bits

Table 7: Parameter regimes for a PRF with $O(\kappa^c)$ total key and evaluation bits, $c \ge 1$, fixed integer d, and fixed $0 < \theta \le 1 - \varepsilon$. All logical words and server cells have width w. The lower bounds apply even in the last row.

environment may choose each request from the earlier traces and logical answers. An online simulator receives only successive leakages and generates the trace, while the ideal RAM functionality supplies the logical responses. Together, the simulated trace and ideal responses give an environment view computationally indistinguishable from its real view. Security covers arbitrary modes and the full API; information that the environment encodes into later public requests is part of the declared leakage. Computational lower bounds additionally require that the exact reference execution, including preprocessing, has polynomial expected probe count, uniformly over the polynomial parameter range; nonuniform advice selects epoch scales from expected costs, and the entropy decoder in the proof need not be efficient (Convention A.4).

Family quantifiers and counted memory. Computational lower bounds have the following quantifiers: for each fixed secure implementation family and fixed polynomial parameter bounds, the inequality holds for all sufficiently large κ, uniformly over parameters satisfying both those polynomial bounds and the theorem's assumptions. The security parameter threshold may depend on the family. The multiplicative lower-bound constants depend only on the fixed constants in the theorem's assumptions. Statistical lower bounds instead hold for all sufficiently large n, with constants and thresholds depending only on the fixed constants in the theorems' assumptions. Upper bounds assert existence of secure parameterized families with the stated costs, for sufficiently large κ and all budgets satisfying the respective theorem's assumptions. Budgets are positive, polynomially bounded, and have polynomial-length descriptions.

Lower bounds permit any fixed constant multiple of the stated memory; upper bounds use $O(m)$ words with fixed constants (which may also depend on θ for the polynomial-client variant). Replacing m by a constant fraction changes H by only a constant factor. For an exact mw-bit cap, apply (25): reserve the stipulated constant fraction for the shared PRF key and evaluator, and choose q as a sufficiently small constant fraction of the remaining word budget. The buffer term then fits with constant slack, and the stash and control terms are $O(w \log_2^2 n)$ and fit in that slack for sufficiently large n. Choose the threshold constant C_0 in Theorem 3.2 after these allocations so that $q = \Theta(m)$ still satisfies the requirement $q \ge \log_2^{12}(2+n)$ from Appendix B; in the polynomial-client variant it still exceeds $n^{\theta/2}$ for all sufficiently large n.

Conventional PRFs and word size. The following instantiation makes the counted-workspace requirement concrete without assuming a compact cryptographic key.

Corollary G.1 (A conventional-PRF instantiation)**.** *Suppose a PRF with a κ-bit key has total key and evaluation workspace $O(\kappa^c)$ bits on the required $O(\log_2 \kappa)$-bit inputs and outputs, for a fixed*

$c \geq 1$. *For any fixed integer* $d \geq 1$, *take*

$$n = \kappa^d, \qquad w = \Theta(\kappa^c), \qquad m = \lceil C_0 \log_2^{12}(2+n) \rceil.$$

For sufficiently large parameters, Theorems 3.1 and 3.2 give matching frontiers with $H = \Theta(\log_2 n)$ *in both budget ranges, with adaptive security for the constructions. Logical words and server cells have the same width* w.

Proof. The PRF uses an $O(1/m) = o(1)$ fraction of the mw client bits. The remaining memory meets the construction threshold, and $m \leq n^{1-\varepsilon}$ for every fixed $0 < \varepsilon < 1$ eventually. □

Parameter regimes. Table 7 lists three concrete regimes for the client memory, the word size, and the security parameter. Suppose the PRF key and peak evaluation workspace total $O(\kappa^c)$ bits on the required $O(\log_2 \kappa)$-bit inputs and outputs. For logarithmic words and $m \geq n^\theta$, choose a fixed integer $d > \max\{1, c/\theta\}$ and set $n = \kappa^d$. Then

$$\frac{O(\kappa^c)}{mw} = O\left(\frac{\kappa^{c-d\theta}}{\log_2 \kappa}\right) = o(1),$$

so a reserved constant fraction of client memory suffices. Addresses, table identifiers, and counters use $O(\log_2 n) = O(\log_2 \kappa)$ bits on each prescribed polynomial horizon.

For polylogarithmic m, Corollary G.1 instead uses $w = \Theta(\kappa^c)$, making the evaluator's memory fraction $O(1/m)$. Combining polylogarithmic m with $w = \Theta(\log_2 n)$ would leave only $\log_2^{O(1)} \kappa$ client bits, too few for a conventional κ-bit key; such a regime would require an explicit compact-key security and workspace guarantee. The exponent 12 was chosen loosely as a sufficient primitive and buffer threshold.

From trace privacy to encrypted storage. The computational constructions also protect a passive server's view of cell contents by a constant-factor wrapper with bounded counters. The masking step is standard counter-mode encryption with a PRF [BDJR97]; the incremental re-encryption sweep described below controls counter size and counted workspace. Preallocate the construction's $S = O(n)$ physical slots, initializing unused slots to zero. Encode each w-bit cell by a key-slot bit, a nonce, and its value masked by a pseudorandom w-bit pad. Keep two fresh masking keys, counted with the table keys, and reuse the same evaluation workspace with a separate purpose domain.

During each sweep, all new writes use the newer key and a fresh counter nonce. After every original probe, read and reencrypt the next physical slot in a fixed cyclic scan under that newer key. After S original probes all slots use it: discard the older key, generate a fresh newer key in its slot, and restart the sweep and nonce counter. A key receives at most $2S$ writes during a sweep, and the initial key receives the S initialization writes, so no nonce is reused within a key's lifetime and the counters always have $O(\log_2 n)$ bits. The key-slot bit selects the retained key when a cell is read. Every access reads or writes its whole $O(1)$-cell encoding. The extra scan contributes only a constant number of probes per original probe, including on unbounded closing tails, and adds $O(1)$ control words besides the counted keys and evaluator. Hence space, initialization, budgets, and individual caps change only by constant factors.

On every polynomial execution, a hybrid over the independently generated masking keys gives fresh write pads independent of both data and table hashes. The server's ciphertext view can then be simulated from the trace by assigning a fresh masked value on each write and returning the stored encoding on reads. The ciphertext simulator is online and maintains its encodings across all

requests, so the same argument applies to the adaptive environment of Definition B.8. Standard adaptive-oracle PRF security gives the computational claim with the same declared leakage.

H Variants of the read frontier

We record two variants of the read frontier: the lower bounds extended to a private read-modify-write interface, and statistical realizations of the constructions on polynomial horizons when a larger client-bit allowance replaces the PRF assumption.

The private-RMW interface. The lower bounds also apply to private read-modify-write (RMW) calls that hide their address and update function, even if the RMW kind is public: compile a read to the identity function and a write of v to the constant function $x \mapsto v$, returning or discarding the old value respectively. This preserves probes, adds $O(w + \log_2 n)$ counted client bits, and changes H only by a constant factor.

H.1 Statistical trace privacy with counted routing

Corollary H.1 (Statistical matching constructions on polynomial horizons with large words)**.** *Retain the polynomial parameter and horizon bounds, word-size upper bound, and client-word assumptions of Theorem 3.2. Replace its PRF assumption by*

$$w \geq \kappa, \qquad mw \geq C_{\mathrm{rt}} n \log_2(2+n), \tag{59}$$

for a sufficiently large fixed constant C_{rt}. Then all schedules of Theorem B.1, the frontier of Theorem 3.2, and the adaptive frontier of Corollary B.10 have realizations that are statistically trace-private on every prescribed polynomial horizon, without a cryptographic assumption. They use at most mw client bits and $O(n)$ server cells. The request-cost bounds stated in those results remain unchanged, as do the separately charged initialization and public-write costs. Correctness error and statistical simulation error are negligible over every prescribed polynomial horizon. Here statistical adaptive simulation means the joint-view guarantee of Definition B.8 with negligible total-variation distance in place of computational indistinguishability, for executions of the prescribed length.

Proof. We replace the PRFs used in Lemma B.2 by explicitly sampled random routing choices, storing all choices that may be reused in counted client memory. In the BigHT, SmallHT, and combined table of [AKL+20, Constructions 7.1, 8.5, and 8.7], a real build-input key x needs only a tuple $(a(x), c(x), o(x), s(x))$: its major-bin choice, the cuckoo candidate pair in that bin's SmallHT, and the candidate pairs for the overflow and combined-stash tables. Each tuple, together with its key, occupies $O(\log_2 n)$ bits. Sample all these values in their original public ranges, independently across keys and outer-table instances, even when some will be unused.

Before building a table of capacity s, a fixed scan of its padded input gathers these keys and generates their tuples. This adds $O(s)$ probes and leaves the input unchanged. Store the key–tuple dictionary locally; its searches require no server probes. Retain each present-key tuple through the key's real lookup, then discard it after the lookup completes. Nonrecurrence guarantees that no later real lookup in that table instance needs the tuple. Tuples for keys never queried are discarded when the table retires. For a key absent from the original build input, generate a fresh tuple for its lookup and discard it afterward; such a real query cannot recur, and a table instance never acquires a new key. Dummy calls use the original procedures' fresh random choices.

To couple this implementation to independent true random functions, first expose each key's major bin. Conditional on that choice, the candidate pair in the selected bin's SmallHT is sampled

independently. A real SmallHT lookup occurs only in that major bin; an overflow hit instead causes a dummy SmallHT lookup in a fresh random bin. For each key x, only the selected bin's SmallHT routing choices are evaluated. The overflow and combined-stash pairs are independent as well. Presampling present-key tuples therefore realizes every function evaluation during the build and the key's sole real lookup exactly. For an absent key, sampling a fresh tuple at its sole real lookup has exactly the same distribution as evaluating independent random functions at that key. Discarding a tuple after that lookup changes no later evaluation, since real queries cannot recur and extraction makes no new keyed evaluations. This argument covers adaptive nonrecurrent queries and fresh table instances with overlapping logical key sets.

Use the same local final stashes, direction padding, and bounded continuations as in Lemma B.2. Sampling integers in public ranges with polynomial caps introduces only negligible error. By Lemma B.2, the per-table failure probability with truly random routing choices is at most $s^2 \exp[-\Omega(\log_2 \kappa \log_2 \log_2 \kappa)]$, which remains negligible under polynomial composition. All other array operations are information theoretic. Thus the merge, shuffle-invariant, and canonical-export proofs, followed by the adaptive simulation proof of Theorem B.9, compose statistically.

At any time the sum of the capacities of live and temporary tables is $O(n)$. The dictionaries therefore occupy $O(n \log_2 n)$ bits, and tuple generation needs $O(\log_2 n)$ additional bits and a constant number of word registers. In the memory ledger (25), replace the shared PRF term by $C'_{\mathrm{rt}} n \log_2(2+n)$ bits and these registers. Choose C_{rt} large enough that this term uses a fixed reserved fraction of mw, and choose the buffer parameter q as a sufficiently small fixed fraction of m. The existing buffers, stashes, counters, and transient workspaces then fit in the remaining bits, with $q = \Theta(m)$ and the exact cap mw respected. No server-space bound changes. The added build scan is linear, so the schedule and tuning proofs retain their original asymptotic costs. □

The statistical privacy guarantee holds on every prescribed polynomial horizon, as in Section 2.2. The statistical lower bounds for the read frontier need only privacy on horizon $N+2$, so this frontier is matched statistically in this horizon-bounded sense.

For example, $n = \kappa$, $w = \kappa$, and $m = C_0 \log_2^{12}(2+n)$ satisfy (59) for sufficiently large κ. The corollary protects the trace defined in Section 2. Applying the cell-content encryption wrapper of Appendix G still requires its cryptographic assumption and its additional counted workspace.

Comparison with external-memory statistical ORAM. Krapivin and Shi [KS26] recently gave a statistically secure optimal all-private ORAM with $O(\log_B N)$ block transfers per request for transfer blocks of $B \geq \log_2^{1+\varepsilon} N$ words. Two differences keep it from settling whether statistical privacy can match the mixed-mode frontier without the additional client-memory requirement $mw \geq C_{\mathrm{rt}} n \log_2(2+n)$ imposed by Corollary H.1. First, its cost unit is the transfer of a large block, whereas our model has logical words and physical cells of equal size and charges every cell probe; our lower and upper bounds are stated in that regime only. Second, all its accesses are private, so it has no public cache, no nonrecurrence obligation created by public reads, and no frequency-dependent split of maintenance between the modes. We leave open whether its techniques yield a statistical matching construction for the mixed-mode frontier without the counted routing allowance (59), with or without large blocks.

I Further comparisons and read-frontier examples

I.1 Further related work

Public-query lower bounds. Patel, Persiano, and Yeo [PPY20] prove logarithmic lower bounds for encrypted multi-maps, including plaintext-query settings, and their public-query proof [PPY20, Section 5.2, Theorem 19] already changes private updates while fixing public queries. Their measure is a single worst-case expected cost normalized by the number of updated or returned values, and a logarithmic bound on it does not preclude constant public-query and logarithmic private-update cost, an arrangement we prove impossible. We bound the two average costs separately, with explicit dependence on the private frequency and budget, and match the resulting frontier.

Lower-bound machinery. Our epochs follow the chronogram method [FS89] and the information-transfer encodings of [PD06], as adapted to ORAM in [LN18, PY19, PY23]; the randomized-endpoint charge is from [PY23, full version, Lemma 4], whose event transfer compares two different queries after the same updates, and thus does not apply to public queries. Komargodski and Lin [KL21] treat general cell sizes; we use only equal word and cell sizes. The bound of [HKKS19] holds even without operation boundaries, which our trace exposes; [WW18] gives barriers to unconditional bounds for read-only ORAM, which our explicit private budget and hidden-kind requirement on $\mathcal{W}_{R,R}$ sidestep; and [BKV24] proves a logarithmic bound for memory checking.

Construction primitives. Our constructions use hierarchical rebuilding [GO96, KLO12], OptORAMa's consuming tables and array primitives [AKL+20], PanORAMa's shuffled inputs [PPRY18], linear deduplication and deamortization [AKLS21, OS97], and an external-memory sorter [GM11]. The new components we introduce are a public cache that prevents repeated hierarchy lookups by public reads, a retiring merge that preserves consuming table nonrecurrence and the joint shuffle invariant, and calendars that confine private maintenance to helpers under separate budgets and caps.

PIR-based mixed-mode ORAM. The scheme of Zhang, Wang, and Weng [ZWW25] uses server computation, as do Onion ORAM [DvDF+16] and ORAM/PIR composition [AFN+17]. It obtains $O(1)$ public and $O(\log_2 n)$ private bandwidth for blocks of $\Omega(b_{\mathrm{PIR}} + \log_2^2 n)$ bits, with semi-honest security for non-adaptively chosen requests in a PIR-hybrid model that discloses read/RMW kinds, and a public RMW that hides its update function. Our server is passive, costs are probes to cells of logical word size, security is adaptive with full-API trace privacy, and public writes disclose their values; we show that in this model the above bandwidth bounds cannot hold together (Corollary 3.3).

Other asymmetric models. Write-only ORAM [BMNO14] assumes reads are unobserved, whereas our server observes both reads and writes, so privacy must also hide connections between public-read probes and earlier private-update probes. Write-public ORAM [AABD26] discloses updates and protects reads, the pattern of $\mathcal{W}_{W,R}$; it improves write bandwidth while keeping the read cost of Path ORAM [SvDS+13], whereas we study joint probe costs and preparation charges under budgets. Their read-public construction [AABD26, Section 6] also gives $O(\log_2 n)$ probe costs for both public reads and private writes. Under our lower-bound assumptions, our read frontier proves this public-read cost optimal at constant private frequency and a logarithmic private budget, and characterizes the dependence on both frequency and budget. Oblivious immutable memory [AH25] instead restricts how cells may be updated.

I.2 Read-frontier examples

Some special cases of the read frontier, valid for both average-input and worst-case-input private budgets, with C a sufficiently large constant and $c, \alpha > 0$ fixed:

Private probability p	Private budget Q	Optimal public-read cost
$\Theta(1)$	CH	$\Theta(H)$
$(\log_2 n)^{-c}$	CH	$\Theta(H/\log_2 H)$
n^{-c}	CH	$\Theta(1)$
$\Theta(1)$	$H^{1+\alpha}$	$\Theta(H/\log_2 H)$
$\Theta(1)$	$(n/(m+1))^{\alpha}$	$\Theta(1)$

Below the threshold $C_1 F_p$, kind disclosure matters. On $\mathcal{W}_{R,W}$ at constant p with a sufficiently large fixed average-input budget, hidden kinds force $R_N \geq (n/(m+1))^{\Omega(1)}$, whereas with disclosed kinds $R_N = O(H)$ is achievable with $\Pi_N = \mathrm{negl}(\kappa)$, because a disclosed private write need not look anything up (Appendix C.1). A worst-case-input private budget, which must also cover all-private inputs, is always $\Omega(H)$.

References

[AABD26] Ananya Appan, James Austgen, Kushal Babel, and Sourav Das. Write-public oblivious RAM and its applications. Cryptology ePrint Archive, Paper 2026/2163, 2026. https://eprint.iacr.org/2026/2163.

[AFN+17] Ittai Abraham, Christopher W. Fletcher, Kartik Nayak, Benny Pinkas, and Ling Ren. Asymptotically tight bounds for composing ORAM with PIR. In *Public-Key Cryptography—PKC 2017, Part I*, volume 10174 of *Lecture Notes in Computer Science*, pages 91–120, 2017. https://doi.org/10.1007/978-3-662-54365-8_5.

[AH25] Ananya Appan and David Heath. Oblivious immutable memory. Cryptology ePrint Archive, Paper 2025/593, 2025. https://eprint.iacr.org/2025/593.

[AKL+20] Gilad Asharov, Ilan Komargodski, Wei-Kai Lin, Kartik Nayak, Enoch Peserico, and Elaine Shi. OptORAMa: Optimal oblivious RAM. In *Advances in Cryptology—EUROCRYPT 2020, Part II*, volume 12106 of *Lecture Notes in Computer Science*, pages 403–432, 2020. Journal version: *Journal of the ACM*, 70(1), Article 4, 2023, https://doi.org/10.1145/3566049. Section, functionality, and theorem numbers in this paper refer to the corrected manuscript dated November 18, 2020 (Cryptology ePrint Archive, Paper 2018/892).

[AKLS21] Gilad Asharov, Ilan Komargodski, Wei-Kai Lin, and Elaine Shi. Oblivious RAM with worst-case logarithmic overhead. In *Advances in Cryptology—CRYPTO 2021, Part IV*, volume 12828 of *Lecture Notes in Computer Science*, pages 610–640, 2021. https://doi.org/10.1007/978-3-030-84259-8_21. Section and theorem numbers refer to the full version (Cryptology ePrint Archive, Paper 2021/1123). Journal version: *Journal of Cryptology*, 36, Article 7, 2023, https://doi.org/10.1007/s00145-023-09447-5.

[AKS83] Miklós Ajtai, János Komlós, and Endre Szemerédi. An $O(n \log n)$ sorting network. In *Proceedings of the Fifteenth Annual ACM Symposium on Theory of Computing*, pages 1–9, 1983. https://doi.org/10.1145/800061.808726.

[BDJR97] Mihir Bellare, Anand Desai, Eron Jokipii, and Phillip Rogaway. A concrete security treatment of symmetric encryption. In *Proceedings of the 38th Annual Symposium on Foundations of Computer Science*, pages 394–403, 1997. Full version: https://web.cs.ucdavis.edu/~rogaway/papers/sym-enc.pdf.

[BKV24] Elette Boyle, Ilan Komargodski, and Neekon Vafa. Memory checking requires logarithmic overhead. In *Proceedings of the 56th Annual ACM Symposium on Theory of Computing (STOC 2024)*, pages 1712–1723, 2024. https://doi.org/10.1145/3618260.3649686.

[BMNO14] Erik-Oliver Blass, Travis Mayberry, Guevara Noubir, and Kaan Onarlioglu. Toward robust hidden volumes using write-only oblivious RAM. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, pages 203–214, 2014. https://doi.org/10.1145/2660267.2660313.

[CT06] Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory*. Wiley, second edition, 2006. https://doi.org/10.1002/047174882X.

[DvDF+16] Srinivas Devadas, Marten van Dijk, Christopher W. Fletcher, Ling Ren, Elaine Shi, and Daniel Wichs. Onion ORAM: A constant bandwidth blowup oblivious RAM. In *Theory of Cryptography—TCC 2016-A, Part II*, volume 9563 of *Lecture Notes in Computer Science*, pages 145–174, 2016.

[FNO21] Brett Hemenway Falk, Daniel Noble, and Rafail Ostrovsky. Alibi: A flaw in cuckoo-hashing based hierarchical ORAM schemes and a solution. In *Advances in Cryptology—EUROCRYPT 2021, Part III*, volume 12698 of *Lecture Notes in Computer Science*, pages 338–369, 2021. https://doi.org/10.1007/978-3-030-77883-5_12. Full version: https://eprint.iacr.org/2020/997.

[FS89] Michael L. Fredman and Michael E. Saks. The cell probe complexity of dynamic data structures. In *Proceedings of the Twenty-First Annual ACM Symposium on Theory of Computing*, pages 345–354, 1989. https://doi.org/10.1145/73007.73040.

[GGM86] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions. *Journal of the ACM*, 33(4):792–807, 1986. https://doi.org/10.1145/6490.6503.

[GM11] Michael T. Goodrich and Michael Mitzenmacher. Privacy-preserving access of outsourced data via oblivious RAM simulation. In *Automata, Languages and Programming—ICALP 2011, Part II*, volume 6756 of *Lecture Notes in Computer Science*, pages 576–587, 2011. Full version: arXiv:1007.1259v2, May 2, 2011, https://arxiv.org/abs/1007.1259v2; we cite Sections 3.2 and 4.2 and Theorem 6 of the full version.

[GO96] Oded Goldreich and Rafail Ostrovsky. Software protection and simulation on oblivious RAMs. *Journal of the ACM*, 43(3):431–473, 1996. https://doi.org/10.1145/233551.233553.

[HKKS19] Pavel Hubáček, Michal Koucký, Karel Král, and Veronika Slívová. Stronger lower bounds for online ORAM. In *Theory of Cryptography—TCC 2019, Part II*, volume 11892 of *Lecture Notes in Computer Science*, pages 264–284, 2019. https://doi.org/10.1007/978-3-030-36033-7_10. Full version: arXiv:1903.03385.

[HRMS20] Steven R. Howard, Aaditya Ramdas, Jon McAuliffe, and Jasjeet Sekhon. Time-uniform Chernoff bounds via nonnegative supermartingales. *Probability Surveys*, 17:257–317, 2020. https://doi.org/10.1214/18-PS321.

[KL21] Ilan Komargodski and Wei-Kai Lin. A logarithmic lower bound for oblivious RAM (for all parameters). In *Advances in Cryptology—CRYPTO 2021, Part IV*, volume 12828 of *Lecture Notes in Computer Science*, pages 579–609, 2021. https://doi.org/10.1007/978-3-030-84259-8_20. Full version: https://eprint.iacr.org/2020/1132. Journal version: *SIAM Journal on Computing*, 54(2):503–544, 2025, https://doi.org/10.1137/21M1428431.

[KLO12] Eyal Kushilevitz, Steve Lu, and Rafail Ostrovsky. On the (in)security of hash-based oblivious RAM and a new balancing scheme. In *Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 143–156, 2012.

[KS26] Andrew Krapivin and Elaine Shi. Optimal statistical oblivious RAM and search tree in the external-memory model. Cryptology ePrint Archive, Paper 2026/2419, 2026. https://eprint.iacr.org/2026/2419.

[LN18] Kasper Green Larsen and Jesper Buus Nielsen. Yes, there is an oblivious RAM lower bound! In *Advances in Cryptology—CRYPTO 2018*, volume 10992 of *Lecture Notes in Computer Science*, pages 523–542, 2018. https://cs.au.dk/~larsen/papers/oram.pdf.

[OS97] Rafail Ostrovsky and Victor Shoup. Private information storage (extended abstract). In *Proceedings of the Twenty-Ninth Annual ACM Symposium on Theory of Computing*, pages 294–303, 1997. https://doi.org/10.1145/258533.258606.

[PD06] Mihai Pătraşcu and Erik D. Demaine. Logarithmic lower bounds in the cell-probe model. *SIAM Journal on Computing*, 35(4):932–963, 2006. https://doi.org/10.1137/S0097539705447256.

[Pes18] Enoch Peserico. Deterministic oblivious distribution (and tight compaction) in linear time. arXiv preprint arXiv:1807.06719, 2018. https://arxiv.org/abs/1807.06719.

[PPRY18] Sarvar Patel, Giuseppe Persiano, Mariana Raykova, and Kevin Yeo. PanORAMa: Oblivious RAM with logarithmic overhead. In *Proceedings of the 59th IEEE Annual Symposium on Foundations of Computer Science*, pages 871–882, 2018. https://doi.org/10.1109/FOCS.2018.00087.

[PPY20] Sarvar Patel, Giuseppe Persiano, and Kevin Yeo. Lower bounds for encrypted multi-maps and searchable encryption in the leakage cell probe model. In *Advances in Cryptology—CRYPTO 2020*, volume 12170 of *Lecture Notes in Computer Science*, pages 433–463, 2020. https://doi.org/10.1007/978-3-030-56784-2_15. Full version: https://eprint.iacr.org/2019/1132.

[PY19] Giuseppe Persiano and Kevin Yeo. Lower bounds for differentially private RAMs. In *Advances in Cryptology—EUROCRYPT 2019*, volume 11476 of *Lecture Notes in Computer Science*, pages 404–434, 2019. Full version: Electronic Colloquium on Computational Complexity, Report TR18-181, 2018, https://eccc.weizmann.ac.il/report/2018/181/.

[PY23] Giuseppe Persiano and Kevin Yeo. Lower bound framework for differentially private and oblivious data structures. In *Advances in Cryptology—EUROCRYPT 2023*, volume 14004 of *Lecture Notes in Computer Science*, pages 487–517, 2023. Full version:

Cryptology ePrint Archive, Paper 2022/1553, revised February 27, 2023. https://eprint.iacr.org/2022/1553.

[SvDS+13] Emil Stefanov, Marten van Dijk, Elaine Shi, Christopher W. Fletcher, Ling Ren, Xiangyao Yu, and Srinivas Devadas. Path ORAM: An extremely simple oblivious RAM protocol. In *Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security*, pages 299–310, 2013. https://doi.org/10.1145/2508859.2516660.

[WW18] Mor Weiss and Daniel Wichs. Is there an oblivious RAM lower bound for online reads? In *Theory of Cryptography—TCC 2018*, volume 11240 of *Lecture Notes in Computer Science*, pages 603–635, 2018.

[ZWW25] Wenhao Zhang, Xiao Wang, and Chenkai Weng. Efficient mixed-mode oblivious RAMs. Preprint, Cryptology ePrint Archive, Paper 2025/1057, June 2025. https://eprint.iacr.org/2025/1057.

AI methodology

We used OpenAI ChatGPT (Codex) and Anthropic Claude (Claude Code) to assist with research development, drafting, and revision; this affected all sections of the paper.

ChatGPT was used to refine our initial ideas for what the mixed-mode lower bounds could be. We generated the proofs with ChatGPT, and then iteratively used it to improve the results, at times guiding it in directions we thought promising.

ChatGPT was then used to generate an initial paper draft from the collected results, and Claude was used to rewrite it for improved readability.

We then manually checked all paper contents and edited them for clarity, readability, and consistency. ChatGPT and Claude were used to fill in some gaps during this editing pass. We have verified the correctness and originality of all final content, including references.