A Lower Bound for One-Round Oblivious RAM

David Cash, Andrew Drucker, and Alexander Hoover(✉)

University of Chicago, Chicago, USA
{davidcash,alexhoover}@uchicago.edu, andy.drucker@gmail.com

Abstract. We initiate a fine-grained study of the round complexity of Oblivious RAM (ORAM). We prove that any one-round balls-in-bins ORAM that does not duplicate balls must have either $\Omega(\sqrt{N})$ bandwidth or $\Omega(\sqrt{N})$ client memory, where N is the number of memory slots being simulated. This shows that such schemes are strictly weaker than general (multi-round) ORAMs or those with server computation, and in particular implies that a one-round version of the original square-root ORAM of Goldreich and Ostrovksy (J. ACM 1996) is optimal. We prove this bound via new techniques that differ from those of Goldreich and Ostrovksy, and of Larsen and Nielsen (CRYPTO 2018), which achieved an $\Omega(\log N)$ bound for balls-in-bins and general multi-round ORAMs respectively. Finally we give a weaker extension of our bound that allows for limited duplication of balls, and also show that our bound extends to multiple-round ORAMs of a restricted form that include the best known constructions.

1 Introduction

Oblivious RAM (ORAM), introduced by Goldreich and Ostrovsky [11], is a primitive for hiding access patterns to an array held by an untrusted party. It is of interest in complexity theory, where one is concerned with the power of oblivious RAM programs which access memory in a manner independent of their inputs, and also for applications like outsourcing encrypted data and protecting secure processors against untrusted memory. ORAM has been studied extensively, with many variants which all in some form define an ORAM to be a stateful, secret-keyed algorithm that provides a client-side interface for reading and writing to an array. The algorithm does not have enough state to store the array itself, so it is allowed to interact with a more powerful but untrusted party that can (e.g. a larger physical memory, or a cloud server). For clarity, we refer to this party as a *server*. Security requires that the addresses being read and written to are hidden from the server.

This work concerns *balls-in-bins* ORAMs, which are a restricted form of ORAM that is powerful enough to capture the best-known (optimal) constructions. At a high level such ORAMs obey two constraints: (1) they interact with a server that acts only as a passive array, accepting read and write requests to cells of the array (below we call such servers *array-only*), and (2) the ORAM treats

R. Pass and K. Pietrzak (Eds.): TCC 2020, LNCS 12550, pp. 457–485, 2020.
https://doi.org/10.1007/978-3-030-64375-1_16

array values as abstract symbols, only moving them from one cell to another[1]. In particular, we do not consider schemes where the server processes data, such as by applying homomorphic encryption.

Intuitively, ORAMs with an array-only server simulate access to a "virtual" array with N_1 cells for the client by reading and writing to a "physical" array with N_2 cells held at the server, for N_2 usually larger than N_1 ($N_2 = \Theta(N_1 \text{polylog} N_1)$ is typical). They typically work by translating one virtual operation into several physical operations, inserting dummies and shuffling real data to hide the intended addresses of the physical operations. The state can be used to hold some values from the array.

ORAM constructions aim to minimize the (bandwidth) *overhead*, which is defined to be the number of physical operations per virtual operation. A very simple (stateless even) ORAM can work by simply storing the N_1 cells in place at the server, and simulating accesses by scanning the entire array at the server, incurring overhead N_1 (here $N_1 = N_2$). While it is not usually explicitly mentioned, another extreme ORAM can use a large state of N_1 array cells to trivially store the virtual array without any server interaction, achieving zero overhead.

Much research on ORAM has targeted more efficient overhead. In their original work, Goldreich and Ostrovsky gave a more advanced construction with $O(\log^3 N_1)$ overhead, and recent work gave a construction with overhead $O(\log N_1)$ [1,22], which is known to be optimal [18] for ORAMs with array-only servers.

Round-Complexity of ORAMs. We initiate the detailed study of the *round complexity* of balls-in-bins ORAMs. It has been observed several times that many of the physical operations (i.e., those processed by the array-only server) of ORAMs can be batched together in parallel rather than issued one-at-a-time, as the ORAM is defined to issue those operations independent of their outcomes. (To be more precise, one generalizes the notion of an array-only server to accept batches of array operations; We fix the details later.) Reducing rounds is desirable for efficiency and simplicity of implementation. But in all efficient constructions there appears to be an inherent limit to this type of batching optimization, as ORAMs adapt some of their physical operations based on the outcome of prior physical operations.

The issue of rounds of general, non-balls-in-bins ORAM, has been considered by Williams and Sion [26] and Garg, Mohassel, and Papamanthou [9], who constructed single-round ORAMs that used server computation (i.e. their server is not array-only). The latter work also noted that both of the families of ORAM schemes with poly-logarithmic bandwidth (hierarchical [1,12,14,17,19,22] and tree-based [5,21,24,25]) had $O(\log N_1)$ round complexity, where N_1 is again the number of cells in the virtual array to be simulated.

[1] Most of our results require that the balls be moved to exactly one location rather than copied, i.e. do not allow for *duplication* of balls.

Our Contributions. This work proves an overhead lower bound for balls-in-bins ORAMs that operate in a single round[2]. It then gives extensions of this result to somewhat more general ORAMs that can store multiple copies of each ball. Finally this work applies the one-round bound to obtain a bound on multi-round balls-in-bins ORAMs of a restricted form that we call "partition-restricted" that captures the best-known bounded-round constructions, showing that they are optimal for ORAMs of this form.

Towards sketching our one-round bound, we observe first that the one-round setting is particularly sensitive to the amount of ORAM state compared to multi-round ORAM. If one is studying $O(1)$-rounds schemes, the state can always be stashed at the server, at the cost of one round, as long as the state size is less than the bandwidth overhead. But in the one-round case (or k-round, for fixed k) we will see that the size of the state is crucially relevant.

Our first main result is an unconditional proof that any one-round balls-in-bins ORAM which does not duplicate balls must either have $\Omega(\sqrt{N_1})$ state or $\Omega(\sqrt{N_1})$ bandwidth overhead. This bound is tight up to logarithmic factors for state, as an optimal construction is a one-round version of the square-root ORAM [11] with $O(\sqrt{N_1}\log N_1)$ state.

Our techniques differ from those of prior ORAM lower bounds, which fall into two categories. The first date back to the original Goldreich and Ostrovsky work, and give bounds on balls-in-bins ORAMs via counting arguments, showing that any particular physical access sequence can only satisfy a bounded number of virtual request sequences. The second comes from a recent line of work initiated by Larsen and Nielsen [18], who proved bounds against general ORAMs via a novel usage of information transfer arguments to show that many consecutive operations must frequently overlap in order to be correct and oblivious.

Our bounds follow intuition similar to the techniques of Larsen and Nielsen, but are for balls-in-bins schemes. At a high level, we show that a one-round requirement and correctness force an ORAM to request overlapping sets of array cells, unless it has $\Omega(\sqrt{N_1})$ client memory or bandwidth. This actually follows from a simple attack but a subtle analysis. Below we first present a simplified version of the bound for ORAM schemes that have almost no client memory (and in particular are only allowed to maintain a program counter). This was the simplest type of ORAM we could find that was non-trivial to bound, and already encapsulates the main difficulties. We then extend our proof to schemes with more client memory. Our version of balls-in-bins schemes does not allow for multiple of copies of balls to be made, but we can give a weaker bound for a bounded number of copies. This latter bound is tight for a constant number of copies, but is loose for larger numbers of copies, becoming trivial if a ball is copied $N_1^{1/4}$ time.

Finally, we sketch how prior ORAMs can be viewed in our formalism for rounds, and show that the square-root ORAM matches our bound. We then

[2] In this work, a "round" is interpreted to be a read request for several cells, followed by a write request to several cells; We discuss the motivation for this below. Being permissive on the notion of a round only makes our lower bounds stronger.

observe that for any constant k, a natural "k-th root" version of that ORAM gives a $(k-1)$-round of a special form with $O(kN_1^{1/k})$ overhead and $O(N_1^{1/k})$ state[3]. While we cannot prove anything non-trivial even for two-round ORAM, we can show that these ORAMs fall into a class of "partition-restricted" ORAMs, and are optimal for that class. The observation is simple: Since these ORAMs predictably access only a relatively small region of memory in their first $(k-2)$ rounds, we can view that region as state and collapse them to one-round schemes to which our one-round bound applies.

In the full version, we additionally consider another restricted class of balls-in-bins ORAM that we call *static*. These ORAMs can not move balls between physical cells on the server after writing them, which seems to not have been considered explicitly previously. Intuitively, such ORAMs can be thought of as "balls-in-bins" PIR schemes, and it is possible that one could hope for a weak type of protection (say, for a bounded number of operations, or with some non-negligible security bound). We observe the counting argument of Goldreich and Ostrovsky easily gives a strong bound for unbounded operation sequences, but that our techniques give a sharper bound for concrete parameters and provide a lower bound for bounded operation sequences. For instance, we show that even if a static ORAM is only required to remain oblivious for N_1+Q+1 operations, it must have overhead or state $\Omega(Q)$, for $Q \leq \sqrt{N_1}$, which follows from proofs similar to our main results. Additionally, we prove that to support an arbitrary number of operations, the ORAM must have overhead or state $\Omega(N_1)$.

Related Work. Goldreich and Ostrovsky were the first to define ORAM and proved the first $\Omega(\log n)$ lower bound for the bandwidth of balls-in-bins ORAMs [11], without any restriction on the number of rounds. Boyle and Naor [2] pointed out some key assumptions in the original proof and asked if they could be overcome. Soon after, Larsen and Nielsen removed the assumptions and obtained the same $\Omega(\log n)$ bound using novel information transfer techniques [18]. After their result, the same bound has been extended with fewer assumptions [15] and to other oblivious data structures [16].

Most of the lower bound work has been on amortized bandwidth and does not consider any restrictions or bounds on round complexity. However, recent work by Chan, Chung, and Shi [3] showed a round lower bound for Oblivious *Parallel* RAM (OPRAM). Showing that any OPRAM must have $\Omega(\log m)$ rounds, where m is the number of processors. OPRAM bounds are distinct from non-parallel ORAM bounds, as they concern the different issue of coordination amongst processors.

Many ORAM constructions have been given in the literature that pay attention to rounds. In their work introducing ORAM, Goldreich and Ostrovsky define a 2-round ORAM as a warm-up for their hierarchical construction [11]. More recently, Goodrich et al. [13] presented a family of constant round ORAM constructions. Several works gave one-round ORAM constructions with server computation [4–10,20,26]. This line of work allows the server holding the data to

[3] One can also obtain a similar construction by modifying tree-based ORAMs [23,24] to use $(k-1)$ levels of recursion.

perform some computation as part of the protocol, rather than the server being an array which can only read and write to requested cells. The previous lower bounds for ORAM do not apply to this model, and neither do ours.

Organization. Section 2 gives definitions. Sections 3, 4, and 5 give our lower bounds for counter-only, general one-round, and multiple-copy schemes respectively. In Sect. 6 we recall the square root construction and its bounded-round variants in our notation, and finally we conclude with a discussion of open problems in Sect. 7.

2 Preliminaries

ORAM Syntax. We give a definition of the ORAM primitive that tailored to the single-round case, and then later extend it to some fixed number of rounds. Our definition most closely follows that of Wang, Chan, and Shi [25], with changes that we discuss below.

We start with an intuitive sketch of Definition 1 below, which is itself quite short. It models a one-round ORAM simulating a virtual array with N_1 cells, with each cell storing a *block* from a set $\mathcal{B}_1$ (e.g. $\mathcal{B}_1 = \{0,1\}^{w_1}$). An ORAM scheme should accept *read operations* (which consist of an address $a \in [N_1]$) and *write operations* (which consist of an address/block pair $(a, d) \in [N_1] \times \mathcal{B}_1$). Correctness requires that in the course of processing a sequence of operations, the last written block written at that address should be returned for read operations (we will formalize this statement later), and obliviousness will require that the addresses a in the sequence are hidden.

The scheme will interact with an array-only server holding a physical array consisting of N_2 cells, each storing a block from the set $\mathcal{B}_2$, which may or may not equal $\mathcal{B}_1$ (parameters with subscripts 1 and 2 will correspond to the virtual array and physical array respectively). The ORAM scheme interacts with the server by sending read and write operations, this time with addresses in $[N_2]$ and blocks in $\mathcal{B}_2$. The server is assumed to always respond correctly. We assume an ORAM comes with associated sets StSp and RSp for the *state space* and *randomness space* respectively. The state space is the set of all possible settings for the data that the ORAM can hold between processing read/operations (so, for example, if $\mathsf{StSp} = \mathcal{B}_1^{N_1}$ then the ORAM can hold the entire virtual array and ignore the server entirely). The randomness space will not be restricted or particularly relevant for quantitative bounds but making it explicit (rather than declaring the ORAM has access to a random tape) fixes a sample space on which every random variable is defined. We remark that secret keys can be sampled (and persistently stored) in the randomness space in addition to any coins that may be used.

Our results require a precise definition of rounds for an ORAM. Intuitively, a round should consist of sending a tuple of read/write operations from the ORAM to the server, which applies the writes and then responds with the results of the read operations. Afterwards, the client updates its local state and continues,

either with more rounds or by replying for the virtual operation (i.e. outputting a block in $\mathcal{B}_1$ in the case of a read, or simply stopping in the case of a write).

We opt for a definition that is somewhat more permissive by defining a round to consist of a tuple of read operations (below specified by Access) followed by a tuple of write operations and a returned block (both below specified by Out; these may depend on what is returned by the read operations). This version of the definition simplifies the accounting for rounds without weakening our lower bounds.

Definition 1. *Let* $\mathcal{B}_1, \mathcal{B}_2, \mathsf{StSp}, \mathsf{RSp}$ *be sets with* $\perp \in \mathsf{StSp}, \perp \notin \mathcal{B}_1$*, and let* N_1, N_2 *be positive integers. For* $j = 1, 2$ *define the sets*

$$\mathsf{RdOps}_j = [N_j], \quad \mathsf{WrOps}_j = [N_j] \times \mathcal{B}_j, \quad \textit{and} \quad \mathsf{Ops}_j = \mathsf{RdOps}_j \cup \mathsf{WrOps}_j.$$

A one-round ORAM scheme *(with respect to* $\mathcal{B}_1, \mathcal{B}_2, N_1, N_2, \mathsf{StSp}, \mathsf{RSp}$*) is a pair of functions* $\mathsf{O} = (\mathsf{Access}, \mathsf{Out})$*,*

$$\begin{aligned} \mathsf{Access} &: \mathsf{Ops}_1 \times \mathsf{StSp} \times \mathsf{RSp} \to \mathsf{RdOps}_2^* \\ \mathsf{Out} &: \mathcal{B}_2^* \times \mathsf{Ops}_1 \times \mathsf{StSp} \times \mathsf{RSp} \to (\mathcal{B}_1 \cup \{\perp\}) \times \mathsf{WrOps}_2^* \times \mathsf{StSp}. \end{aligned}$$

This models the following usage: A sample from RSp (e.g. keys and a random tape) is chosen and then kept private at the client, and the state is initialized to a canonical value $\perp \in \mathsf{StSp}$. The function Access takes as input a requested virtual operation along with the current state and the randomness, and outputs a list of physical read operations on the server memory. The function Out takes the results of these operations (i.e. the blocks from read operations), the virtual operation being requested, and the state and randomness. Its first output is the result of the operation (either the block resulting from a read, or $\perp$ for a write). Its second output is a set of write operations that should be applied at the server. When we use Out in the games defined in Fig. 1, we write elements in WrOps_2^* as $(\mathsf{wrts}, \mathbf{d}_w)$, which denote the ordered sets of locations and data to write respectively. Finally, Out also outputs an updated state, in preparation for the next operation.

As mentioned before the definition, this syntax is actually somewhat stronger than one-round, since the client is allowed defer its writes until after it sees the results of the reads. The definition of Wang et al. makes a similar choice, where the ORAM is allowed to "piggyback" its interaction with the server between operations, receiving the next operation before being required to output the result of the previous one [25]. Our bounds apply to either model but we found ours simpler. Finally we remark that allowing Access to update the state is unnecessary, as Out gets all of the information available to it.

ORAM Correctness and Obliviousness. We next define correctness and obliviousness of an ORAM scheme. In both cases, every definition we are aware of only explicitly considered non-adaptive definitions, where an adversary chooses operations all at once. We give adaptive definitions, and note that standard arguments can separate the adaptive and non-adaptive versions. Our bounds

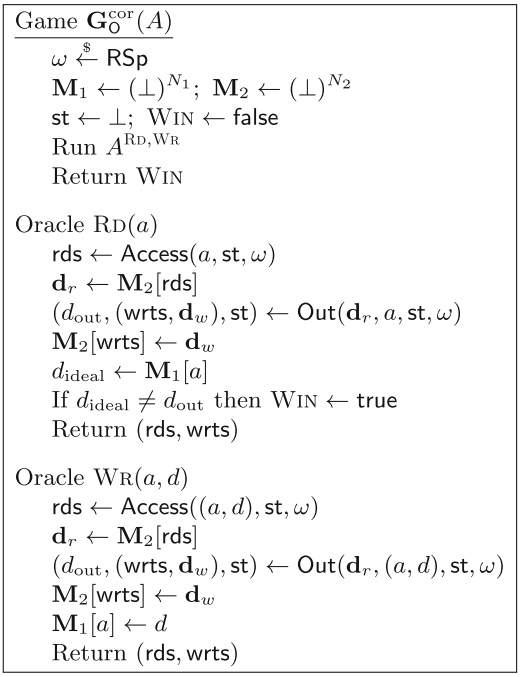

Fig. 1. Game $\mathbf{G}^{\mathrm{cor}}_{\mathsf{O}}$ for an ORAM scheme $\mathsf{O} = (\mathsf{Access}, \mathsf{Out})$.

will ultimately only need a non-adaptive adversary and thus be stronger, but practical constructions should likely aim for the stronger definition.

The correctness definition uses game $\mathbf{G}^{\mathrm{cor}}_{\mathsf{O}}(A)$ from Fig. 1, which we sketch now. At a high level, it allows the adversary to adaptively request that virtual operations be run, and gets to see the physical addresses touched. The adversary wins if it ever catches the ORAM returning an incorrect block on a read operation.

This game starts by choosing an element of randomness space, and initializes two arrays: $\mathbf{M}_1$ with N_1 cells, and $\mathbf{M}_2$ with N_2 cells. The first array will model the "ideal" virtual array that should be maintained in the course of operation, and the second will hold the physical array that the server would maintain. An initial state is fixed, and the adversary is given access to two oracles, and attempts to trigger a "win" flag.

The first oracle accepts a virtual read operation $a \in \mathsf{RdOps}_1$, and the game processes the query by running Access and Out on the appropriate inputs, updating $\mathbf{M}_2$ as a real server would. It also performs the "ideal" virtual read operation on $\mathbf{M}_1$, and sets the win flag if the ideal output differs from what the ORAM output. Finally it returns the addresses from the read and write operations, simulating what a server would see.

The second oracle is similar but processes write operations. It applies the correct write to the ideal array $\mathbf{M}_1$, and also simulates the ORAM running with physical array $\mathbf{M}_2$. It also returns the addresses of the physical operations.

Definition 2. *Let* $\mathsf{O} = (\mathsf{Access}, \mathsf{Out})$ *be a one-round ORAM scheme with respect to* $\mathcal{B}_1, \mathcal{B}_2, N_1, N_2, \mathsf{StSp}, \mathsf{RSp}$*, and let* A *be an adversary. The* correctness advantage of A against O *is defined to be*

$$\mathbf{Adv}^{\mathrm{cor}}_{\mathsf{O}}(A) = \Pr[\mathbf{G}^{\mathrm{cor}}_{\mathsf{O}}(A) = 1],$$

where $\mathbf{G}^{\mathrm{cor}}_{\mathsf{O}}$ *is defined in Fig. 1. We say that* O *is* perfectly correct *if this advantage is zero for any adversary* A*.*

Game $\mathbf{G}^{\text{obl-}b}_{\mathsf{O}}(A)$
- $\omega \xleftarrow{\$} \mathsf{RSp}$
- $\mathbf{M}_2 \leftarrow (\perp)^{N_2}$
- $\mathsf{st} \leftarrow \perp$
- $b' \leftarrow A^{\textsc{Rd},\textsc{Wr}}$
- Return b'

Oracle $\textsc{Rd}(a_0, a_1)$
- $\mathsf{rds} \leftarrow \mathsf{Access}(a_b, \mathsf{st}, \omega)$
- $\mathbf{d}_r \leftarrow \mathbf{M}_2[\mathsf{rds}]$
- $(d_{\mathrm{out}}, (\mathsf{wrts}, \mathbf{d}_w), \mathsf{st}) \leftarrow \mathsf{Out}(\mathbf{d}_r, a_b, \mathsf{st}, \omega)$
- $\mathbf{M}_2[\mathsf{wrts}] \leftarrow \mathbf{d}_w$
- Return $(\mathsf{rds}, \mathsf{wrts})$

Oracle $\textsc{Wr}((a_0, d_0), (a_1, d_1))$
- $\mathsf{rds} \leftarrow \mathsf{Access}((a_b, d_b), \mathsf{st}, \omega)$
- $\mathbf{d}_r \leftarrow \mathbf{M}_2[\mathsf{rds}]$
- $(d_{\mathrm{out}}, (\mathsf{wrts}, \mathbf{d}_w), \mathsf{st}) \leftarrow \mathsf{Out}(\mathbf{d}_r, (a_b, d_b), \mathsf{st}, \omega)$
- $\mathbf{M}_2[\mathsf{wrts}] \leftarrow \mathbf{d}_w$
- Return $(\mathsf{rds}, \mathsf{wrts})$

Fig. 2. Games $\mathbf{G}^{\text{obl-}b}_{\mathsf{O}}$, $b = 0, 1$, for an ORAM scheme $\mathsf{O} = (\mathsf{Access}, \mathsf{Out})$.

The obliviousness definition uses games $\mathbf{G}^{\text{obl-}b}_{\mathsf{O}}(A)$, $b = 0, 1$, from Fig. 2. These are left-right indistinguishability games, where the adversary can now query its oracles with two operations (either both read or both writes). The oracle processes one of the operations, updating a physical array $\mathbf{M}_2$, and returns the physical addresses touched, modeling what a curious server would see.

Definition 3. *Let* $\mathsf{O} = (\mathsf{Access}, \mathsf{Out})$ *be a one-round ORAM scheme with respect to* $\mathcal{B}_1, \mathcal{B}_2, N_1, N_2, \mathsf{StSp}, \mathsf{RSp}$*, and let* A *be an adversary. The* obliviousness advantage of A against O *is defined to be*

$$\mathbf{Adv}^{\mathrm{obl}}_{\mathsf{O}}(A) = \Pr[\mathbf{G}^{\text{obl-}1}_{\mathsf{O}}(A) = 1] - \Pr[\mathbf{G}^{\text{obl-}0}_{\mathsf{O}}(A) = 1].$$

We say that O *is* perfectly oblivious *if this advantage is zero for any adversary* A*.*

In the obliviousness definition the data written to the physical array is not revealed to the distinguishing adversary. Standard encryption can be applied to upgrade a scheme to a model where data is also hidden. This definition also reveals to the adversary if operations are reads or writes, and implicitly when one operation ends and the next begins (see Hubácek et al. [15], which considered models where this distinction is not revealed).

ORAM RESOURCE MEASURES. We will be interested in the *overhead* and *state size* of an ORAM. We will consider worst-case and amortized overhead.

Definition 4. *Let* $\mathsf{O} = (\mathsf{Access}, \mathsf{Out})$ *be a one-round ORAM with respect to* $\mathcal{B}_1, \mathcal{B}_2, N_1, N_2, \mathsf{StSp}, \mathsf{RSp}$. *We say that* O *has* worst-case overhead p *if* Access *and* Out *always output at most* p *operations. We say that* O *has* amortized overhead p *if for every* $Q \geq 0$ *and every adversary* A *issuing* Q *queries in* $\mathbf{G}^{\mathrm{cor}}_{\mathsf{O}}$, *the total the number of operations returned in oracle queries is at most* pQ *with probability* 1.

We define the state size of O *to be* $\log |\mathsf{StSp}|$.

BALLS-IN-BINS ORAM. Our results will only apply to a restricted class of schemes that handle memory in a symbolic "balls-in-bins" manner. This was originally informally defined by Goldreich and Ostrovsky, and we follow most closely the definition of Boyle and Naor [2].

Definition 5. *Let* $\mathsf{O} = (\mathsf{Access}, \mathsf{Out})$ *be a one-round ORAM with respect to* $\mathcal{B}_1, \mathcal{B}_2, N_1, N_2, \mathsf{StSp}, \mathsf{RSp}$. *We say that* O *is* balls-in-bins *if it is of the following special form:*

- $\mathcal{B}_2$ *is the disjoint union of* $\mathcal{B}_1$ *and a set of bitstrings* $\{0,1\}^{w_2}$. *We call the members of* $\mathcal{B}_1$ balls.
- StSp *has the form* $\{0,1\}^m \times (\mathcal{B}_1 \cup \{\perp\})^r$. *That is, a state of* O *consists of* m *bits along with an array of* r *balls/*$\perp$ *entries. For a state* $\mathsf{st} = (\sigma, \mathbf{reg})$, *the entries in* $\mathbf{reg}$ *are called* registers.
- *The function* Out *satisfies the following:*
  If $\mathsf{Out}(\mathbf{d}_r, (a, d), \mathsf{st}, \omega) = (d_{\mathrm{out}}, \mathsf{wrts}, \mathbf{d}_w, \mathsf{st}')$, *where* $\mathsf{st} = (\sigma, \mathbf{reg})$ *and* $\mathsf{st}' = (\sigma', \mathbf{reg}')$, *then*
  - $\mathbf{reg}'$ *and* $\mathbf{d}_w$ *are formed by moving* d *and the balls from* $\mathbf{reg}$ *and* $\mathbf{d}_r$, *and then populating their remaining entries with arbitrary non-ball values. (Any ball may be moved to at most one place.)*
  - d_{out} *appears in* $\mathbf{d}_r$ *or* $\mathbf{reg}$.

Intuitively, this definition requires that whenever the ORAM returns a block for a read, the history of that block can be traced back to when it is written, as at each step the ORAM can only move the balls between physical cells and/or registers.

We note that this definition does not allow for copying a ball multiple times, and our main bound does not hold if such copies are allowed. In Sect. 5 we give a relaxed definition and prove a weaker bound in the presence of duplicate balls.

Our warm-up bound will consider even more restricted balls-in-bins ORAMs that maintains almost no state. Restricting the scheme to no state at all is not

interesting, as then it cannot even vary its requests as they are repeated. Thus we define a *counter only* scheme to maintain only a program counter of the number of operations performed.

Definition 6. *We say that a one-round ORAM* O *is* counter-only *if it satisfies all of the conditions for a balls-in-bins scheme, except that it has* $\mathsf{StSp} = \{0,1\}^*$ *(i.e. no registers), and its state at all times is a simple counter of the number operations run (initialized to zero, and then incremented on each run of* Out*).*

We remark that a counter-only scheme can still have a secret key (say a PRF key, or even a random function), which is modeled in the randomness space. Giving the ORAM a counter allows it to change its operations as time progresses, and non-trivial constructions are possible. For us it has the advantage of forcing the ORAM to behave in a simple combinatorial manner, as at each step the possible physical cells accessed for each operation are fixed once the randomness is fixed.

3 Warm-Up: Lower Bound for Counter-Only Schemes

We first give a bound for the restricted case of counter-only schemes with perfect correctness and perfect obliviousness, and in the next section remove all of these restrictions.

Theorem 1. *Let* $\mathsf{O} = (\mathsf{Access}, \mathsf{Out})$ *be a counter-only one-round balls-in-bins ORAM scheme with respect to* $\mathcal{B}_1, \mathcal{B}_2, N_1, N_2, \mathsf{StSp}, \mathsf{RSp}$*. Assume* $|\mathcal{B}_1| \geq N_1$*. Suppose* O *is perfectly correct, perfectly oblivious, and has worst-case overhead* p*. Then*

$$p \geq C\sqrt{N_1},$$

where C *is an absolute constant.*

Proof. For concreteness we prove the theorem with $C = 0.1$. Let O have the syntax from the theorem, and assume it is perfectly correct and $p < C\sqrt{N_1}$. We construct a non-adaptive randomized adversary A and show that O cannot be perfectly oblivious, i.e. that $\mathbf{Adv}^{\mathrm{obl}}_{\mathsf{O}}(A) > 0$. The adversary works as follows:

1. For $i = 1, \ldots, N_1$, query $\textsc{Wr}((i, b_i), (i, b_i))$, where $b_1, \ldots, b_{N_1}$ are arbitrary distinct balls from $\mathcal{B}_1$. Ignore the responses.
2. Let $T = \sqrt{N_1}$. Choose $J \xleftarrow{\$} [N_1]^T$, a sequence of T i.i.d. uniform virtual addresses, and query

$$\textsc{Rd}(J[1], J[1]), \ldots, \textsc{Rd}(J[T], J[T]).$$

   Let $\mathsf{rds}_1, \ldots, \mathsf{rds}_T \subseteq [N_2]$ be the physical addresses read for each query.
3. Choose $t \xleftarrow{\$} [T]$, set $j_0^* \leftarrow J[t]$, and $j_1^* \xleftarrow{\$} [N_1]$. Query $\textsc{Rd}(j_0^*, j_1^*)$ and let rds^* be the physical addresses read.
4. Output 0 if there exists an address $a \in \mathsf{rds}^*$ that also appears in rds_t but not in any of $\mathsf{rds}_1, \ldots, \mathsf{rds}_{t-1}$. Otherwise output 1.

We claim that

$$\Pr[\mathbf{G}_{\mathsf{O}}^{\text{obl-}0}(A) = 1] \leq 0.2 \tag{1}$$

and

$$\Pr[\mathbf{G}_{\mathsf{O}}^{\text{obl-}1}(A) = 1] \geq 0.9 \tag{2}$$

which together will prove the theorem.

We start with the latter inequality (2), which is intuitively simple; it follows because the read operation for j_1^* can only overlap in the required way (meaning at a "fresh" physical address that was not previously touched) with p of the previous reads, and the random variable t is chosen independently of these overlaps. Formally, condition on ω, J and j_1^*; then t (which is still used in the final step) remains uniform. The set rds^* can overlap at a point a in the required way with at most p of the sets $\mathsf{rds}_1, \ldots, \mathsf{rds}_T$. Thus the probability the adversary will output 0 is bounded by $p/T \leq 0.1$.

Proving (1) is more subtle. We sketch our approach before giving the formal proof. Our plan is to focus on the starting physical position of the "test" ball $b^* = b_{J[t]}$ after step 1 of the adversary, and argue that with good probability this position will work as the address a in step 4, that is, it is accessed for the first time at query t in step 2, and then again in step 3.

To argue that this position is touched for the first time at query t in step 2, we use a counting argument. Since $p < C\sqrt{N_1}$, at most $p(t-1) < CN_1$ balls in total could have been touched in the $t-1$ prior operations. Thus most balls are untouched, remaining where they started. We are picking one at random and thus have a good probability of accessing the starting position of b^* for the first time in the t-th query.

More difficult is arguing that the starting position of b^* is touched again in 3. A counting argument no longer works for b^*, because now b^* *was* previously touched with probability 1 (it is no longer independent), and the ORAM has a chance to move it. At this point perfect correctness and the assumption that O is counter-only combine to come to the rescue. Note that since O is counter-only, once ω and b^* have been chosen, the locations read in step 3 are fixed, independent of the "history" in step 2. The crucial observation is that the starting location of b^* must be read in step 3 if there is *any* history that would leave b^* in its starting place. This is due to perfect correctness, since the ORAM must be correct for that history, even if it is not the one that actually happened! All that remains is to apply another counting argument showing that most balls have a history in which they do not move, and the combine (via a union bound) with the argument about step 2.

Now for the formal proof of (1). We will prove this holds conditioned on any fixed ω, t, and $J[1], \ldots, J[t-1]$; The only remaining choices are $J[t], \ldots, J[T]$, which are still uniform. By our assumption that O is balls-in-bins and has no registers, after the first stage of the adversary we have that every ball $b_1, \ldots, b_{N_1}$ lies in exactly one entry of $\mathbf{M}_2$; Let $q_1, \ldots, q_{N_1} \in [N_2]$ be their respective indices. We will show that with probability at least 0.8 in the conditional space, $a = q_{J[t]}$

satisfies the conditions for outputting 0 in the final step of the adversary. This establishes that 1 is output with probability at most 0.2 in this game.

We do this in two steps, following the sketch. We write $q^* = q_{J[t]}$ for the index of b^* index. We first show that

$$\Pr[q^* \in \mathsf{rds}_t \setminus \bigcup_{k=1}^{t-1} \mathsf{rds}_k] \geq 0.9 \tag{3}$$

and then that

$$\Pr[q^* \in \mathsf{rds}^*] \geq 0.9. \tag{4}$$

(In both cases, the probability is over $J[t] \in [N_1]$ only, the latter because the construction is counter-only.) A union bound gives the claimed 0.8 probability.

We proceed with the first step. Since $J[1], \ldots, J[t-1]$ and ω are fixed, the sets $\mathsf{rds}_1, \ldots, \mathsf{rds}_{t-1}$ are also fixed. We have

$$\Pr[q^* \notin \bigcup_{k=1}^{t-1} \mathsf{rds}_k] \geq 1 - \frac{(t-1)p}{N_1} \geq 0.9,$$

because $J[t]$ is uniform in the conditional space and q^* is thus uniform on a set of size N_1, while the union of the rds_k is of size at most $(t-1)p$. By the perfect correctness and balls-in-bin assumptions on O, we must have that $q^* \in \mathsf{rds}_t$ whenever q^* is not in any of $\mathsf{rds}_1, \ldots, \mathsf{rds}_{t-1}$, because ball b^* will still reside at index q^* of $\mathbf{M}_2$. Thus the event in the probability is actually equivalent to $q^* \in \mathsf{rds}_t \setminus \bigcup_{k=1}^{t-1} \mathsf{rds}_k$, and we have completed (3), the first step in proving (1).

We now prove the second step (4). The argument from the first step does not apply, because the test ball is being read twice (once in the second stage, and then again at the third stage of the adversary). Instead, here we will apply the assumption that O is counter-only and one round (so far everything we have proved would hold with small modifications even if O were an arbitrary multi-round scheme).

The set rds^* is computed by $\mathsf{Access}(J[t], \mathsf{st}, \omega)$ where $\mathsf{st} = N_1 + T + 1$ is the counter. The key observation is that this set must contain q^* if there exists *any* value $\hat{J} \in [N_1]^T$ such that $q^* \notin \bigcup_{k=1}^{T} \mathsf{Access}(\hat{J}[k], N_1 + k, \omega)$. This is true because after these accesses ball b^* would not be touched and hence still reside at index q^*. Thus q^* must be touched by $\mathsf{Access}(J[t], \mathsf{st}, \omega)$ (as O is perfectly correct) in case it has not moved. (Note we have used that O is counter-only here; If it had more state, then the set $\mathsf{Access}(J[t], \mathsf{st}, \omega)$ could change based on the "history", but it can not change when O is counter-only.)

Thus we only need to lower-bound the number of values of $J[t]$ for which there exists $\hat{J} \in [N_1]^T$ such that $q^* \notin \bigcup_{k=1}^{T} \mathsf{Access}(\hat{J}[k], k + N_1, \omega)$. This is easy: Just take some arbitrary choice of $\hat{J}$. The union of their access sets will have size at most $pT \leq 0.1N_1$, so we get that there are $0.9N_1$ values for $J[t]$ will work. This establishes (4) and (1). □

4 Lower Bound for General Balls-in-Bins Schemes

We extend the previous theorem to general balls-in-bins schemes. The step form the previous proof that falls apart is (4), which relied on the final "test" access issued by the scheme to be independently of the request history. This no longer holds when the scheme has state beyond a counter, and indeed state can enable an ORAM to sometimes avoid the repeated test index.

The previous strategy can be made to work even with state. Intuitively, the scheme will not be able to remember "too much" of the history, and so its bounded state can only help avoid the test with a relatively small advantage. We formalize this intuition by bounding, for any state, the number of histories for which a particular state can be used to evade the attack, and ultimately union bound over all possible states.

Theorem 2. *Let* $\mathsf{O} = (\mathsf{Access}, \mathsf{Out})$ *be a one-round balls-in-bins ORAM scheme with respect to* $\mathcal{B}_1, \mathcal{B}_2, N_1, N_2, \mathsf{StSp}, \mathsf{RSp}$. *Assume* $|\mathcal{B}_1| \geq N_1 \geq 10^6$. *Suppose* O *has worst-case overhead* $1 \leq p < C\sqrt{N_1}$ *and state size* s *and for every adversary* A, $\mathbf{Adv}^{\mathrm{cor}}_{\mathsf{O}}(A) < 0.001$ *and* $\mathbf{Adv}^{\mathrm{obl}}_{\mathsf{O}}(A) < 0.4$. *Then*

$$ps \geq CN_1,$$

where C *is an absolute constant.*

Before giving the proof, we note that this bound is tight up to logarithmic factors for constructions with $p = O(\sqrt{N_1})$, with the matching construction being a modification of the "square-root ORAM" that we recall in Sect. 6. We leave open to determine the optimal state size for constructions with larger p. We also note that $\mathsf{StSp} = \{0,1\}^m \times (\mathcal{B}_1 \cup \{\perp\})^r$ for balls-in-bins ORAM, and for the following proof, we need only assume $m + r \log N_1 < 0.001 N_1/p$ for a contradiction, which is slightly stronger than the stated result.

Proof. The proof proceeds as before, with the same adversary A, except we have it issue $T = 0.001N_1/p$ queries in the second stage. We will show that, if $p < 0.001\sqrt{N_1}$ and $s < 0.0001N_1/p$, then

$$\Pr[\mathbf{G}^{\text{obl-0}}_{\mathsf{O}}(A) = 1] \leq 0.55 \tag{5}$$

and

$$\Pr[\mathbf{G}^{\text{obl-1}}_{\mathsf{O}}(A) = 1] \geq 0.999. \tag{6}$$

The bound (6) is proved exactly as before, so we only need to establish (5). We do so via the same strategy, proving analogues of (3) and (4). Throughout the proof, we assume $\mathsf{StSp} = \{0,1\}^m \times (\mathcal{B}_1 \cup \{\perp\})^r$ because the ORAM is balls-in-bins.

Let $\mathsf{rds}_1, \ldots, \mathsf{rds}_T$ and q^* be defined as before. Then an analogue of (3) holds via a very similar proof; In fact we have

$$\Pr[q^* \in \mathsf{rds}_t \setminus \bigcup_{k=1}^{t-1} \mathsf{rds}_k] \geq 0.997 \tag{7}$$

with our parameters now. The only modification to the argument is we must subtract the correctness error 0.001 and also the probability that the test ball is in one of the registers in StSp. By assumption, $r \leq 0.001N_1$, which gives the bound above.

Thus, proving the theorem is reduced to proving an analogue of (4). Specifically, we prove that

$$\Pr[q^* \in \mathsf{rds}^*] \geq 0.5. \tag{8}$$

Combining (7) and (8) via a union bound establishes (5), showing that O will output 0 with at least probability 0.45.

We now prove (8). This requires analyzing how many balls the ORAM can move from their starting positions while maintaining correctness, so we begin with some definitions to quantify this. We define a function $B(\hat{J}, \hat{\omega})$ which takes as input a tuple $\hat{J} \in [N_1]^T$ and $\hat{\omega} \in \mathsf{RSp}$, and counts the number of balls in $\hat{J}$ that will move during the second stage of the adversary (these are the "bad" balls for our attack). Formally, $B(\hat{J}, \hat{\omega})$ works as follows:

1. Run the game $\mathbf{G}_{\mathsf{O}}^{\text{obl-0}}(A)$ with $\omega = \hat{\omega}$, until the end of the first stage. At this point, every ball is either in a unique position in $\mathbf{M}_2$, or in a register. Let $q_1, \ldots, q_{N_1}$ be the indexes of the balls in $\mathbf{M}_2$ or $\perp$ if the corresponding ball is in a register.
2. Continue the game, now also using $J = \hat{J}$ until the end of the second stage of the adversary. Let st be the state of O.
3. Output the number of $j \in \hat{J}$ such that $q_j \notin \mathsf{Access}(j, \mathsf{st}, \hat{\omega})$. (This includes j for which $q_j = \perp$.)

We also define related functions:

- $B(\hat{J}, \hat{\omega}, \hat{\mathsf{st}})$ that is exactly the same as B, except it uses the input state $\hat{\mathsf{st}}$ in step 3 instead of the state computed in step 2.
- $B_{\text{all}}(\hat{J}, \hat{\omega})$ that is exactly the same as B, except for the last step, in which case it outputs the count of $j \in [N_1]$ that satisfy the condition (and not just the $j \in \hat{J}$).
- $B_{\text{all}}(\hat{J}, \hat{\omega}, \hat{\mathsf{st}})$ that is B_{all}, except modified to use $\hat{\mathsf{st}}$ as the state in step 3. This function does not depend on $\hat{J}$, as it can be computed by running step 1, and then skipping to step 3.

The latter three functions will be useful for counting the total number of balls that move, not just those in $\hat{J}$ (in the case of B_{all}). The versions with a hard-coded state $\hat{\mathsf{st}}$ will be useful for steps in the proof where we want to argue about the existence of a good state.

It suffices to show that

$$\Pr_{J,\omega}[B(J, \omega) > 0.25T] \leq 1/5. \tag{9}$$

Assuming this, we have

$$\begin{aligned}\Pr_{J,\omega,t}[q^* \notin \mathsf{rds}^*] &\leq \Pr_{J,\omega,t}[q^* \notin \mathsf{rds}^* | B(J,w) \leq 0.25T] \\ &\quad + \Pr_{J,\omega,t}[q^* \notin \mathsf{rds}^* \wedge B(J,w) > 0.25T] \\ &\leq 1/4 + \Pr_{J,\omega}[B(J,w) > 0.25T] \leq 1/4 + 1/5 < 1/2.\end{aligned}$$

We now prove (9). Our strategy is to condition on whether or not $B_{\mathrm{all}}(J,\omega)$ is large and handle the cases separately. We have that $\Pr_{J,\omega}[B(J,\omega) > 0.25T]$ is at most

$$\Pr_{J,\omega}[B_{\mathrm{all}}(J,\omega) > 0.03N_1] + \Pr_{J,\omega}[B(J,\omega) > 0.25T \wedge B_{\mathrm{all}}(J,\omega) \leq 0.03N_1]. \quad (10)$$

The first term is bounded using Markov's inequality. We assert that for any fixed $\hat{J} \in [N_1]^T$,

$$\mathbb{E}_{\omega}[B_{\mathrm{all}}(\hat{J},\omega)] \leq r + pT + \varepsilon N_1 \leq 0.003N_1,$$

where $\varepsilon = \mathbf{Adv}^{\mathrm{cor}}_{\mathsf{O}}(A)$. This expectation is over ω only. This follows because B will count at most r balls from registers, pT balls moved during the second stage, and (in expectation) at most εN_1 balls on which O errs with our adversary. Each of these contribute at most $0.001N_1$ to the expectation. By Markov's inequality, we get that first term of (10) is at most 0.1.

We complete the proof by bounding the second term of (10). For this, we are aiming to show that, that O is unlikely to enter a state where not too many balls have been moved in total and yet many balls from J have been moved. The challenge is that the state depends on J. We will show that any particular state cannot be useful for too many J, and then take a union bound over all states; It is (only) here that use the fact that O does not have a large state space. Intuitively, without such a bound on the state space, the state st, which depends on J, could be chosen so that $B_{\mathrm{all}}(J,\omega) \leq 0.03N_1$ and yet still $B(J,\omega) > 0.25T$, because $0.25T \ll 0.03N_1$.

Formally, we bound the second term for every fixed $\hat{\omega}$. We observe that it is at most

$$\Pr_{J}[\exists \hat{\mathsf{st}} \in \mathsf{StSp} : B(J,\hat{\omega},\hat{\mathsf{st}}) > 0.25T \wedge B_{\mathrm{all}}(J,\hat{\omega},\hat{\mathsf{st}}) \leq 0.03N_1],$$

where we have used the versions of B and B_{all} with a hard-coded state as input. We then union bound over $\hat{\mathsf{st}} \in \mathsf{StSp}$, so this probability is at most

$$\sum_{\hat{\mathsf{st}} \in \mathsf{StSp}} \Pr_{J}[B(J,\hat{\omega},\hat{\mathsf{st}}) > 0.25T \wedge B_{\mathrm{all}}(J,\hat{\omega},\hat{\mathsf{st}}) \leq 0.03N_1].$$

For a fixed $\hat{\mathsf{st}}$, the probability is at most the chance that at least $0.25T$ of the i.i.d. uniform entries of J land in a pre-determined set of size at most $0.03N_1$

(since ω and st are fixed, $B_{\text{all}}(J, \omega, \mathsf{st})$ is fixed, counting this set, as it does not depend on J). If we denote by X the number of such entries, we have

$$\Pr[X > 0.25T] \leq \Pr[X > 0.03(1 + 7.33)T].$$

By a Chernoff bound this probability is at most

$$\left(\frac{e^{7.33}}{(8.33)^{8.33}}\right)^{0.03T} \leq 0.75^T.$$

Summing over $\hat{\mathsf{st}} \in \mathsf{StSp}$ gives

$$|\mathsf{StSp}| \cdot 0.75^T \leq 2^{0.0001N_1/p} 0.75^{0.001N_1/p} < 0.1$$

for $N_1 \geq 10^6$, because $p < 0.001\sqrt{N_1}$. This completes the bound of the second term of (10). Combining with the bound on the first term completes the proof, giving (9), as desired. □

4.1 Bound for ORAMs with Amortized Overhead

Theorem 2 only applies to ORAMs with worse-cast overhead, but the ideas extend easily to ORAMs with only amortized overhead. As-is, the attack from the previous theorem cannot handle an amortized adversary; For example, the final test read could have exceptionally high overhead, which would allow for the test set to overlap with many of the previous sets. To work around this, our high-level approach is to have the adversary repeat the reading stage of the attack many times and then choose one at random to test for overlaps. An averaging argument shows that with high probability over this random choice, the chosen stage will not have too much overhead and thus the previous reasoning will apply.

Theorem 3. *Let* $\mathsf{O} = (\mathsf{Access}, \mathsf{Out})$ *be a one-round balls-in-bins ORAM scheme with respect to* $\mathcal{B}_1, \mathcal{B}_2, N_1, N_2, \mathsf{StSp}, \mathsf{RSp}$. *Assume* $|\mathcal{B}_1| \geq N_1 \geq 30 \cdot 10^6$. *Suppose* O *has amortized overhead* $1 \leq p < C\sqrt{N_1}$, *state size* s, *and for every adversary* A, $\mathbf{Adv}^{\text{cor}}_{\mathsf{O}}(A) < 0.001$ *and* $\mathbf{Adv}^{\text{obl}}_{\mathsf{O}}(A) < 0.15$. *Then*

$$ps \geq CN_1,$$

where C *is an absolute constant.*

Proof. We will take $C = 0.001/30$ so that most calculations remain similar to the previous proof. Define an adversary A as follows:

1. For $i = 1, \ldots, N_1$, query $\textsc{Wr}((i, b_i), (i, b_i))$, where $b_1, \ldots, b_{N_1}$ are arbitrary distinct balls from $\mathcal{B}_1$. Ignore the responses.
2. Let $T = CN/p$. For $k = 1, \ldots, N_1$, repeat the following:
   (a) Let $J_k \xleftarrow{\$} [N_1]^T$ and query

$$\textsc{Rd}(J_k[1], J_k[1]), \ldots, \textsc{Rd}(J_k[T], J_k[T]).$$

   Call the sets of physical cells accessed $\mathsf{rds}^k_1, \ldots, \mathsf{rds}^k_T$.

(b) Choose $t_k, t'_k \xleftarrow{\$} [T]$ and $J'_k \xleftarrow{\$} [N_1]^T$. Query

$$\text{RD}(J'_k[1], J'_k[1]), \ldots, \text{RD}(J_k[t_k], J'_k[t'_k]), \ldots, \text{RD}(J'_k[T], J'_k[T]).$$

This is a sequence reading J'_k (on both the left and right), except on one random query, namely the t'_k-th query. There, the attack is using a random entry from J_k on the left as a test. We call the set of physical addresses returned by this operation rds^*_k.

3. Choose $i \xleftarrow{\$} [N_1]$. Output 0 if there exists an address $a \in \mathsf{rds}^*_i$ that also appears in rds^i_t but not in any of $\mathsf{rds}^i_1, \ldots, \mathsf{rds}^i_{t-1}$. Otherwise output 1.

This adversary is based on the same idea as in the two previous proofs. The only differences are that it copies the attack N_1 times and only tests one at random. Notice that this adversary always queries $N_1 + 2TN_1 < 3TN_1$ queries. By the definition of amortized overhead, this means less than $3pTN_1$ operations can be returned across the entire sequence.

Throughout the proof, we use notation $J, J', t, t', \mathsf{rds}_1, \ldots, \mathsf{rds}_T, \mathsf{rds}^*$ for the respective variables at the chosen "test window" i to avoid cluttered indices. The rest of the proof will not need to refer to those values with other indices $k \neq i$.

From here we proceed as the previous proof. Assume $p < C\sqrt{N}$ and $s < 0.1CN_1/p$. We will prove

$$\Pr[\mathbf{G}^{\text{obl-0}}_{\mathsf{O}}(A) = 1] \leq 0.7 \tag{11}$$

and

$$\Pr[\mathbf{G}^{\text{obl-1}}_{\mathsf{O}}(A) = 1] \geq 0.85. \tag{12}$$

We begin showing (12). Assume everything is fixed but i, t, t'. Then,

$$\begin{aligned}\Pr_{i,t,t'}[\mathbf{G}^{\text{obl-1}}_{\mathsf{O}}(A) = 0] &\leq \Pr_{i,t,t'}[|\mathsf{rds}^*| \geq 30p] + \Pr_{i,t,t'}[\mathbf{G}^{\text{obl-1}}_{\mathsf{O}}(A) = 0 || \mathsf{rds}^*| < 30p] \\ &\leq 0.1 + 30p/T \leq 0.15.\end{aligned}$$

The final inequality comes because if $\Pr_{i,t'}[|\mathsf{rds}^*| \geq 30p] > 0.1$, then there are $0.1TN_1$ sets with size at least $30p$, which means the total overhead is at least $3pTN_1 > (2T+1)pN_1$.

Now, we move on to prove (11). We will use the same technique to extend the original proof. First, we will use the same notation defining q^* as as the location of the tested ball in the chosen internal i. Then, we will show

$$\Pr[q^* \in \mathsf{rds}_t \setminus \bigcup_{k=1}^{t-1} \mathsf{rds}_k] \geq 0.8. \tag{13}$$

This follows from a similar argument as before. Define $\varepsilon = \mathbf{Adv}^{\mathrm{cor}}_{\mathsf{O}}(A) < 0.001$.

$$\begin{aligned}\Pr[q^* \notin \mathsf{rds}_t \setminus \bigcup_{k=1}^{t-1} \mathsf{rds}_k] &\leq \Pr[q^* \in \bigcup_{k=1}^{t-1} \mathsf{rds}_k | \left|\bigcup_{k=1}^{t-1} \mathsf{rds}_k\right| < 30Tp] \\ &\quad + \Pr[\left|\bigcup_{k=1}^{t-1} \mathsf{rds}_k\right| \geq 30Tp] + \frac{r}{N_1} + \varepsilon N_1 \\ &\leq \frac{30pT}{N_1} + 0.1 + 0.001 + 0.001 \leq 0.15.\end{aligned}$$

Otherwise, at least $0.1N_1$ of the repeated attacks would access a total of $3N_1Tp$, which gives a contradiction.

The final part of the previous proof which must be extended is

$$\Pr[q^* \in \mathsf{rds}^*] \geq 0.45.$$

We extend this claim by considering the expectation and probabilities over i in the same way. We have to redefine and extend all the functions based on $B(J, \omega)$ to follow the query pattern of our new adversary. The new functions also must take new inputs i, t', which specify where to stop running and count the balls, exactly analogous to how the adversary chooses where to plant the repeated read. The positions of the balls will now be marked at the beginning of each attack and the functions will count using those positions given i.

The claims will still be true with these analogous definitions except, we must show, for all fixed $\hat{J} \in [N_1]^{2N_1T}$, then with probability 0.9 over the uniformly random choice of i,

$$\mathbb{E}_{\omega,t'}[B_{\mathrm{all}}(\hat{J}, \omega, i, t')] \leq r + 30pT + \varepsilon N_1 \leq 0.003N_1.$$

As has been establish, with probability 0.9 at most $30pT$ balls accessed in any attack interval. Assuming this, the expectation must be at most $0.003N_1$ or else the ORAM will be incorrect with probability more than 0.001 against an adversary in this interval.

Once this is established, we take all other probabilities assuming this expectation use a union bound. This achieves the bound

$$\Pr[B(J, \omega, i, t') > 0.25T] \leq 0.3,$$

which implies,

$$\Pr[q^* \notin \mathsf{rds}^*] \leq 0.55.$$

This concludes the proof of (11), because we output 0 with probability at least $1 - 0.55 - 0.15 = 0.3$. □

5 Lower Bound for Balls-in-Bins Schemes with Duplicates

The techniques used for the previous proof can be extended to allow the ORAM scheme to have up to D copies of any ball. We start by defining precisely how

such an ORAM is allowed to copy balls, and then we extend our previous proof idea to such ORAM schemes.

Current constructions do not make use of duplication. However, it could be an avenue to achieve low overhead for constant round schemes in principle. We prove a lower bound using our same techniques from the previous sections and show for constant duplication, we achieve a similar bound for one-round ORAM.

Unfortunately, our techniques do not give tight bounds against high duplication. For example, our bound is trivial for ORAM copying a single ball $N_1^{0.25}$ times. In our proof technique, we attempt to force the ORAM to overlap two reads on a specific physical address in a special way. When a ball can be located in many locations, the ORAM can often avoid this behavior by accessing the locations of other copies.

Definition 7. *Let* $\mathsf{O} = (\mathsf{Access}, \mathsf{Out})$ *be a one-round balls-in-bins ORAM with respect to* $\mathcal{B}_1, \mathcal{B}_2, N_1, N_2, \mathsf{StSp} = \{0,1\}^m \times (\mathcal{B}_1 \cup \{\perp\})^r, \mathsf{RSp}$*, except that we relax the balls-in-bins restriction to allow* Out *to copy balls to multiple locations.*

For a deterministic adversary A *in* $\mathbf{G}_{\mathsf{O}}^{\text{obl-0}}$*,* $\mathbf{G}_{\mathsf{O}}^{\text{obl-1}}$*, or* $\mathbf{G}_{\mathsf{O}}^{\text{cor}}$ *and for every* $b \in \mathcal{B}_1$*, after* A *is finished querying its oracles, define*

$$Q_b(A) = \{i \mid \mathbf{M}_2[i] = b\}$$

and

$$R_b(A) = \{i \mid \mathbf{reg}[i] = b\},$$

where $\mathbf{M}_2$ *is the final server memory state in the game, and* $\mathbf{reg}$ *is the final register state of* O*.*

We say O *is* D-duplicate *if for all adversaries* A *which query* $\textsc{Wr}$ N_1 *times with* N_1 *unique balls in* $\mathbf{G}_{\mathsf{O}}^{\text{obl-0}}$*,* $\mathbf{G}_{\mathsf{O}}^{\text{obl-1}}$*, or* $\mathbf{G}_{\mathsf{O}}^{\text{cor}}$*, and for all* $b \in \mathcal{B}_1$

$$|Q_b(A)| + |R_b(A)| \leq D.$$

Theorem 4. *Let* $\mathsf{O} = (\mathsf{Access}, \mathsf{Out})$ *be a one-round* D*-duplicate balls-in-bins ORAM scheme with respect to* $\mathcal{B}_1, \mathcal{B}_2, N_1, N_2, \mathsf{StSp}, \mathsf{RSp}$*. Assume* $|\mathcal{B}_1| \geq N_1 \geq 10^6$ *and* $1 \leq D < 0.5\sqrt{N_1}$*. Suppose* O *has worst-case overhead* $0 < p < C\sqrt{N}/D^2$*, state size* s*, and for every adversary* A*,* $\mathbf{Adv}_{\mathsf{O}}^{\text{cor}}(A) < 0.001/D$ *and* $\mathbf{Adv}_{\mathsf{O}}^{\text{obl}}(A) < 0.4$*. Then*

$$ps \geq CN_1/D^3,$$

where C *is an absolute constant.*

Proof. This proof proceeds in the same structure as before. Assume for a contradiction that $p < 0.001\sqrt{N_1}/D^2$ and $s < 0.0001N_1/(D^3 p)$. We construct an adversary A that works as follows:

1. For $i = 1, \ldots, N_1$, query $\textsc{Wr}((i, b_i), (i, b_i))$, where $b_1, \ldots, b_{N_1}$ are arbitrary distinct balls from $\mathcal{B}_1$. Ignore the responses.
2. Let $T = 0.001N_1/(D^2 p) \geq \sqrt{N_1}$. Choose $J \xleftarrow{\$} [N_1]^{DT}$, a sequence of DT i.i.d. uniform virtual addresses. For $i = 1, \ldots, D$, choose $t_i \xleftarrow{\$} [(i-1)T+1, \ldots, iT]$. Define $j_0^* = J[t_1]$.

3. For $k = 1, \ldots, DT$, if $k = t_i$ for some i then query

$$\text{RD}(j_0^*, J[k]),$$

and otherwise query

$$\text{RD}(J[k], J[k]).$$

Let $\mathsf{rds}_1, \ldots, \mathsf{rds}_{DT} \subseteq [N_2]$ be the physical addresses read for each query.

4. Let $j_1^* \xleftarrow{\$} [N_1]$ and $t_{D+1} = TD + 1$. Query $\text{RD}(j_0^*, j_1^*)$ and let $\mathsf{rds}_{t_{D+1}}$ be the addresses read.
5. Output 0 if there exists a pair of indices $i, j \in [D+1]$ with $i < j$ and an address $a \in \mathsf{rds}_{t_j}$ that also appears in rds_{t_i} but not in any of $\mathsf{rds}_1, \ldots, \mathsf{rds}_{t_i - 1}$. Otherwise output 1.

This attack follows a similar structure as those outlined in previous sections. However, it accesses the targeted ball $D + 1$ times total. Intuitively, we are showing that each of these accesses must touch one of the D initial locations of the balls. If that is true, then there is some pair which accesses the same location by the pigeonhole principle. This pair is identified be the adversary with high probability and that gives our advantage.

We claim that

$$\Pr[\mathbf{G}_{\mathsf{O}}^{\text{obl-0}}(A) = 1] \leq 0.55 \tag{14}$$

and

$$\Pr[\mathbf{G}_{\mathsf{O}}^{\text{obl-1}}(A) = 1] \geq 0.999 \tag{15}$$

which prove the theorem.

We prove similar claims to the previous proofs, but will need to make a pigeonhole argument as well. For a fixed random string ω, by definition O is has at most D duplicates of any ball and has no registers, after the first stage of the adversary we have that every ball $b_1, \ldots, b_{N_1}$ lies in at most D entry of $\mathbf{M}_2$; Let $Q_1, Q_2, \ldots, Q_{N_1} \subseteq [N_2]$ be the sets of their respective indices, each with size at most D.

We begin by proving (14). First, we show

$$\Pr[|Q_{j_0^*} \cap \mathsf{rds}_{t_1}| \geq 1] \geq 0.997, \tag{16}$$

which follows from arguments used in previous proofs. There are at most pT accesses before t_1, only r registers, and O can error on only ε fraction of the inputs. Therefore, over the random choice of t_1, which is independent from previous accesses. None of the at most D balls were touched prior to this access with probability at most $D(pT + r + \varepsilon N_1)/N_1 \leq 0.003$.

We will prove that for every $2 \leq i \leq D + 1$,

$$\Pr[|Q_{j_0^*} \cap \mathsf{rds}_{t_i}| \geq 1] \geq 1 - \frac{0.5}{D} \tag{17}$$

which together with (16) proves that there is some index q^* that lies in two different reads t_i and t_j with probability 0.497 using a union bound.

Then, given this q^*, i and j exists, we will prove, for the smallest pair $i < j$ with the desired overlap,

$$\Pr[q^* \in \mathsf{rds}_{t_i} \setminus \bigcup_{k=1}^{t_i-1} \mathsf{rds}_k] \geq 0.997, \tag{18}$$

which finishes the proof for Eq. (14).

Now we shift our focus to prove Eq. (17) which requires the proof techniques used in the previous extension. We redefine the function $B(\hat{J}, \hat{\omega})$ for this new setting, so it can take in a sequence of variable length $\hat{J} \in [N_1]^{\leq DT}$ and $\hat{\omega} \in \mathsf{RSp}$. Let $\hat{J}$ be length k, then B works as follows:

1. Run the game $\mathbf{G}_{\mathsf{O}}^{\text{obl-0}}(A)$ with $\omega = \hat{\omega}$, $J = \hat{J} \times \perp^{DT-k}$, until the end of the first stage. At this point, every ball is in some set of indices in $\mathbf{M}_2$, or in a register. Let $Q_1, \ldots, Q_{N_1}$ be the sets of indices of the balls $b_1, \ldots, b_{N_1}$ respectively in $\mathbf{M}_2$.
2. Continue the game for another k queries (i.e. $\hat{J}$ is finished). Let st be the state of O.
3. Output the number of $j \in \hat{J}$ such that $|Q_j \cap \mathsf{Access}(j, \mathsf{st}, \omega')| = 0$.

Similarly, we redefine $B(\hat{J}, \hat{\omega}, \hat{\mathsf{st}})$, $B_{\text{all}}(\hat{J}, \hat{\omega})$, and $B_{\text{all}}(\hat{J}, \hat{\omega}, \hat{\mathsf{st}})$ with the updated check condition at the end. Even with this redefinition, it suffices to show, for every $i \geq 2$,

$$\Pr_{J,\omega}[B(J, \omega) > 0.25t_i] \leq 0.2/D. \tag{19}$$

Assuming this, we have for any fixed $i \geq 2$,

$$\begin{aligned}
\Pr_{J,\omega,t}[|Q_{j_0^*} \cap \mathsf{rds}_i| = 0] &\leq \Pr_{J,\omega,t}[|Q_{j_0^*} \cap \mathsf{rds}_i| = 0 \wedge B(J, w) \leq 0.25t_i] \\
&\quad + \Pr_{J,\omega,t}[|Q_{j_0^*} \cap \mathsf{rds}_i| = 0 \wedge B(J, w) > 0.25t_i] \\
&\leq 0.25/D + \Pr_{J,\omega}[B(J, w) > 0.25t_i] \\
&\leq 0.25/D + 0.2/D < 0.5/D.
\end{aligned}$$

In this probability, we note that J is taken according to the distribution that A submits to to the left part of its oracles up to rds_{t_i} which is independently random outside of the locations $t_1, \ldots, t_{i-1}$.

First, we bound (19), by conditioning on the size of $B_{\text{all}}(J, \omega)$. We have that $\Pr_{J,\omega}[B(J, \omega) > 0.25t_i]$ is at most

$$\Pr_{J,\omega}[B_{\text{all}}(J, \omega) > 0.03N_1] + \Pr_{J,\omega}[B(J, \omega) > 0.25t_i \wedge B_{\text{all}}(J, \omega) \leq 0.03N_1]. \tag{20}$$

The first term is bounded using Markov's inequality. We assert that for any fixed $\hat{J} \in [N_1]^{\leq DT}$,

$$\mathbb{E}_{\omega}[B_{\text{all}}(\hat{J}, \omega)] \leq r + pDT + \varepsilon N_1 \leq 0.003N_1/D,$$

where $\varepsilon = \mathbf{Adv}^{\mathrm{cor}}_{\mathsf{O}}(A)$. Just as before, this follows because B will count at most r balls from registers, pDT balls moved during the second stage, and (in expectation) at most εN_1 balls on which O errs with our adversary. Each of these contribute at most $0.001N_1/D$ to expectation. By Markov's inequality, we get that first term of (20) is at most $0.1/D$.

We complete the proof by bounding the second term of (20), in a similar way to before. Intuitively, since the entries of J are distributed according to A, for a set $\hat{\mathsf{st}}$ this is the probability that at least $0.25t_i$ of its entries land in a predetermined set of size at most $0.03N_1$, or equivalently a tail bound on flipping a biased coin $t_i - i$ times. We subtract i, because $i - 1$ values are the same. So, long as 0.25 of the remaining values are covered by $B(J, \omega)$, then 0.25 of all the values will be covered, giving us an upper bound. Formally, upper bound with an existential quantifier over the state and union bound as in the previous section, but we omit the details here.

Take the probability of heads to be 0.03, and let X be the total number of heads seen after t_i independent coin flips. Then, we have, for a fixed $\hat{\omega}$ and fixed $\hat{\mathsf{st}}$,

$$\begin{aligned}\Pr_J[B(J, \hat{\omega}, \hat{\mathsf{st}}) > 0.25(t_i - i)] &\leq \Pr[X > 0.25(t_i - D)] \\ &\leq \Pr[X > (1 + 7.33)0.03(t_i - D)].\end{aligned}$$

Using a Chernoff bound, this probability is at most

$$\left(\frac{e^{7.33}}{(8.33)^{8.33}}\right)^{0.03(t_i - D)} \leq 0.75^{(t_i - D)} \leq 0.75^{T-D}.$$

Then, a union bound of all states gives us our final requirement,

$$|\mathsf{StSp}| \cdot 0.75^{(T-D)} \leq 2^{0.0001N_1/(D^3 p)} 0.75^{0.001N_1/(D^2 p) - D} < 0.1$$

when $D < 0.5\sqrt{N_1}$ and $N_1 \geq 10^6$. This concludes the proof of (17).

We show (18) next. Fix q^*, i and j as before. Then,

$$\Pr[q^* \in \mathsf{rds}_{t_i} \setminus \bigcup_{k=1}^{t_i - 1} \mathsf{rds}_k] \geq 1 - \varepsilon - \frac{(p + r) \cdot (t_i - 1)}{N_1} \geq 0.997$$

because O can error on a most ε faction of the inputs, and there are at most $(p+r)\cdot(t_i - 1)$ balls touched before t_i is read. Also, t_i and thus q^* is uniform and independent from all other reads except for t_k when $k < i$. However, if $q^* \in t_k$ from some k, then we would have taken t_k and t_i as the pair to fix instead. Together with (17) this proves (14).

To prove (15), we condition on ω, J and j_1^*, then the each of the sets rds_{t_j} can overlap with at most p of the $(j-1)T$ previous sets in the desired way. Summing over all possible endpoints shows the probability of outputting 0 is bounded by $Dp/T < 0.001$, which proves 1 is output with probability at least 0.999, completing the proof of the theorem. □

6 Constant-Round ORAM

We define k-round ORAM in our notation and then review (within our formalism) the "square-root construction" given in the original paper on Oblivious RAM by Goldreich and Ostrovsky [11]. We will also present an $O(kN_1^{1/k})$-overhead construction using k-rounds which can be seen as a middle ground between the square-root and hierarchical constructions. A similar construction was given by Goodreich et al. [13], which explores constant round ORAM as an extension of the square root construction for all constants. However, the number of rounds is less explicit than the construction we present.

We then prove a simple corollary of Theorem 2, which shows the constant-round $O(kN_1^{1-1/k})$-overhead constructions are optimal up to logarithmic factors for a restricted class of ORAM we call "partition-restricted" ORAM. This restriction requires that the reads of all rounds except the last fall into a relatively small, pre-determined zone of physical memory. We then note that the given constant-round constructions have this property, but that it does not extend to logarithmic round constructions which do not respect this restriction. This corollary suggests that to achieve better overhead performance for constant rounds, would require new techniques in ORAM constructions.

Constant round ORAM definitions. The k-round definition we give is a natural extension of the one-round definition. We aim for a simple and permissive definition, so we allow the ORAM to issue a sequence of k reads. After each read, the results are accumulated before the final round, which produces the writes and the operation output. We note that allowing writes in the intervening rounds would not strengthen the ORAM, as they can always be deferred without increasing bandwidth in our model.

We remark that other definitions are not typically so permissive. In practice, one would need to store the read results in the ORAM memory which often needs to be small.

Definition 8. *Let $\mathcal{B}_1, \mathcal{B}_2, \mathsf{RSp}, \mathsf{StSp}$ be sets, and N_1, N_2 be positive integers. For $j = 1, 2$ define the sets*

$$\mathsf{RdOps}_j = [N_j], \quad \mathsf{WrOps}_j = [N_j] \times \mathcal{B}_j, \quad \textit{and} \quad \mathsf{Ops}_j = \mathsf{RdOps}_j \cup \mathsf{WrOps}_j.$$

A k-round ORAM scheme (with respect to $\mathcal{B}_1, \mathcal{B}_2, N_1, N_2, \mathsf{StSp}, \mathsf{RSp}$) is a tuple of functions $\mathsf{O} = (\mathsf{Access}_1, \ldots, \mathsf{Access}_k, \mathsf{Out})$,

$$\begin{aligned} \mathsf{Access}_i &: \mathcal{B}_2^* \times \mathsf{Ops}_1 \times \mathsf{StSp} \times \mathsf{RSp} \to \mathsf{RdOps}_2^* \qquad (i = 1, \ldots, k) \\ \mathsf{Out} &: \mathcal{B}_2^* \times \mathsf{Ops}_1 \times \mathsf{StSp} \times \mathsf{RSp} \to (\mathcal{B}_1 \cup \{\perp\}) \times \mathsf{WrOps}_2^* \times \mathsf{StSp}. \end{aligned}$$

We next adapt the correctness and obliviousness definitions to constant-round ORAM. We use the definitions and their associated games as-is, except that in the games we redefine the notation Access to mean the following algorithm, for $\mathsf{op} \in \mathsf{Ops}_1, \mathsf{st} \in \mathsf{StSp}, \omega \in \mathsf{RSp}$:

$\underline{\mathsf{Access}(\mathsf{op}, \mathsf{st}, \omega)}$
$\quad \mathbf{d}_r \leftarrow \perp$
$\quad$ For $i = 1, \ldots, k$:
$\quad\quad \mathsf{rds} \leftarrow \mathsf{rds} \cup \mathsf{Access}_i(\mathbf{d}_r, \mathsf{op}, \mathsf{st}, \omega)$
$\quad\quad \mathbf{d}_r \leftarrow \mathbf{M}_2[\mathsf{rds}]$
$\quad$ Return rds.

This models the accumulated reads mentioned above, where each Access_i gets to see the output of reads for $\mathsf{Access}_1, \ldots, \mathsf{Access}_{i-1}$. The games then provide Out with *all* of the accumulated read results, exactly as specified in their code. The rest of the games are exactly the same.

The state size of a k-round ORAM is measured exactly as before. For worst-case and amortized overhead, we use the same definitions, but with the version of Access defined above.

A version of the Square-Root ORAM. The square-root construction of Goldreich and Ostrovsky is usually described as a multi-round ORAM with no state. Here we show that it can be viewed as an amortized one-round scheme with larger state that matches our lower bounds. Below we extend this to a family of constant-round schemes. As the ideas are very standard in the ORAM literature, we omit the full details.

The ORAM works with an arbitrary set of balls $\mathcal{B}_1$ and virtual memory size N_1, and with physical memory of $N_2 = N_1 + \sqrt{N_1}$ cells with $\mathcal{B}_2 = \mathcal{B}_1$. The randomness space is defined so that an unbounded sequence of random permutations π on $[N_2]$ can be generated[4]. The state of the ORAM consists of a counter $\mathsf{st}.c$ (initially 0, and always between 0 and $\sqrt{N_1}$) and a tuple $\mathsf{st}.\mathsf{Cache}$ of at most $\sqrt{N_1}$ virtual-address/ball pairs.

The ORAM maintains the physical array to hold the N_1 balls at physical addresses $\pi(1), \ldots, \pi(N_1)$, with virtual address a stored at physical address $\pi(a)$, where π is the current random permutation. The physical addresses $\pi(N_1 + 1), \ldots, \pi(N_1 + \sqrt{N_1})$ will be "dummies", which are accessed to cover for when the same virtual address been accesses multiple times. The ORAM stores in $\mathsf{st}.\mathsf{Cache}$ the virtual-address/ball pairs involved in the most recent $\sqrt{N_1}$ operations. To process a read operation, if the requested virtual address a is not in the cache, then ORAM accesses the ball at physical address $\pi(a)$. If on the other hand a is stored in the cache, then the ORAM accesses the next dummy, namely $\pi(N_1 + \mathsf{st}.c)$. After retrieval, balls are held in the cache. After $\sqrt{N_1}$ operations, the cache may be full, so the ORAM downloads the entire physical memory, samples a fresh π, and places the balls in the physical memory according to π.

This ORAM is perfectly oblivious: Independent of the addresses, the ORAM will access random distinct physical addresses for at most $\sqrt{N_1}$ reads (or no addresses for writes), followed by a reads and writes to all N_2 physical cells. It

[4] To be totally rigorous in our formalism, one needs to give the ORAM the ability to remember which permutation π in the sequence is being used, e.g. by providing an unbounded counter that does not count as state.

has amortized overhead $p = (\sqrt{N_1} + (N_1 + \sqrt{N_1}))/N_1 = O(\sqrt{N_1})$ and a state with $m = \log N_1$ bits and $r = \sqrt{N_1}$ registers, making it tight for Theorem 2 up to logarithmic factors.

k^{th}-root ORAM Construction. The ideas in the square-root ORAM generalize to give a $k-1$-round construction with amortized overhead $O(kN_1^{1/k})$ and state size $O(N_1^{1/k})$. This construction is simply a re-parameterization of the well-known hierarchical ORAM of Goldreich and Ostrovsky [11], adjusted to a constant number of levels, so we only sketch the construction, assuming their construction is familiar.

The ORAM holds in its state a cache containing at most $N^{1/k}$ virtual-address/ball pairs. At the physical memory, it maintains $k-1$ "levels", which are regions of physical memory. Level i consists of $O(\log(\frac{1}{\varepsilon})N_1^{(i+1)/k})$ cells storing a hash table capable of holding $N_1^{(i+1)/k}$ balls, except with probability ε, which we consider an independent error parameter. Thus the final $(k-1)$-th layer can hold N_1 balls.

An access happens over $k-1$ rounds. Initially it the ORAM checks the cache, and remembers if the requested virtual address is found or not. Then in the i-th round, the hash table on level i is to be accessed. If the ball has not yet been found, then the table is accessed at the points determined by the hash function for that level. If the ball has been found then a dummy is accessed. Eventually the ball is found and added to the cache (and in the case of writes the ORAM just add them directly).

Eventually the cache will overflow, so the ORAM periodically rebuilds the hash tables according to a schedule that also ensures none of the levels overflow. Namely, after $N_1^{i/k}$ operations, levels $1, \ldots, i$ are downloaded and all of the balls they contain are stored in a rebuilt table on level i. (In our setting we again avoid the complexity of using oblivious sorts; We allow ORAMs to simply to rebuild locally and upload the tables.)

This completes the sketch of the k^{th}-root ORAM. It has state size $O(N_1^{1/k})$ and overhead $O(kN_1^{1/k})$. We can calculate the overhead by observing that after $N_1^{i/k}$ operations, the ORAM performs a rebuild requiring $O(N_1^{(i+1)/k})$ operations. Thus after N_1 operations, this type of rebuild will accumulate a total cost of $O(N_1^{1-i/k} \cdot N_1^{(i+1)/k}) = O(N_1^{1+1/k})$ physical operations. This amortizes to $O(N_1^{1/k})$ overhead, and summing over k gives $O(kN_1^{1/k})$.

Bound for Restricted k-round ORAM. We now partially address the question of whether the k^{th}-root ORAMs are optimal. Our one-round bounds of course do not apply, and adapting them appears to be non-trivial. Instead, we observe that these ORAMs obey a simple restricted property, and then prove that the k^{th}-root ORAM is optimal amongst multi-round ORAMs with this property.

We call this property *partition-restricted*. Intuitively, a multi-round ORAM is *ℓ-partition-restricted* if all of its rounds always access some predetermined regions of ℓ physical cells. For example, the k^{th}-root ORAM is ℓ-partition-restricted for

$\ell = O(N_1^{1-1/k})$, as the first $k-2$ rounds will access tables of that size or less (recall the k^{th}-root ORAM has $k-1$ rounds total).

For such ORAMs we make a simple observation: One can move the physical memory of the first $(k-2)$ rounds into the state of the ORAM, and transform it into a one-round ORAM to which our bound applies.

Definition 9. *Let* $\mathsf{O} = (\mathsf{Access}, \mathsf{Out})$ *be a* k*-round ORAM scheme with respect to* $\mathcal{B}_1, \mathcal{B}_2, N_1, N_2, \mathsf{StSp} = \{0,1\}^m \times (\mathcal{B}_1 \cup \{\perp\})^r, \mathsf{RSp}$ *We say that* O *is* ℓ-partition-restricted *if there exists a set* $P \subseteq [N_2]$ *of size at most* ℓ *such that for every input* $(\mathbf{d}_r, \mathsf{op}, \mathsf{st}, \omega)$ *and* $i = 1, \ldots, k-1$ *we have* $\mathsf{Access}_i(\mathbf{d}_r, \mathsf{op}, \mathsf{st}, \omega) \subseteq P$.

We now show that ℓ-partition-restricted multi-round ORAMs reduce to one-round ORAMs.

Corollary 1. *Let* $\mathsf{O} = (\mathsf{Access}, \mathsf{Out})$ *be a* k*-round balls-in-bins ORAM scheme with respect to* $\mathcal{B}_1, \mathcal{B}_2, N_1, N_2, \mathsf{StSp}, \mathsf{RSp}$. *Assume* $|\mathcal{B}_1| \geq N_1 \geq 30 \cdot 10^6$ *and* $\mathcal{B}_2 = \mathcal{B}_1 \cup \{\perp\}$. *Suppose* O *has amortized overhead* $1 \leq p < C\sqrt{N_1}$, *state size* s, O *is partition-restricted to* ℓ *server cells, and for every adversary* A, $\mathbf{Adv}^{\text{cor}}_{\mathsf{O}}(A) < 0.001$ *and* $\mathbf{Adv}^{\text{obl}}_{\mathsf{O}}(A) < 0.15$. *Then*

$$p(s + \ell \log N_1) \geq C N_1,$$

where C *is an absolute constant.*

This corollary proves that the k^{th}-root is optimal up to logarithmic factors for this restricted class of ORAM. It is notable that this bound is actually independent of the number of rounds the ORAM uses. It only requires that all but the final access are restricted. This means the registers of the client can be outsourced on the server and read as an additional round. So, we assume there are no registers in StSp and achieve the same bound.

Proof. Assume for a contradiction that $(s + \ell \log N_1) < CN_1/p$. Then, we can construct a one-round ORAM O' with state space $\mathsf{StSp}' = \mathsf{StSp} \times (\mathcal{B}_1 \cup \{\perp\})^\ell$. Since O is partition-restricted to ℓ server cells there is a set P which can capture the first $k-1$ access. The new ORAM O' simulates O but whenever O reads or writes to the set P, O' simulates this by reading or writing to the ℓ extra registers in StSp'. Because the first $k-1$ accesses will always read from P, O' only requires accessing the server to simulate the final access, making it one-round.

Notice that $\max_A \mathbf{Adv}^{\text{cor}}_{\mathsf{O}'}(A) \leq \max_A \mathbf{Adv}^{\text{cor}}_{\mathsf{O}}(A)$ and $\max_A \mathbf{Adv}^{\text{obl}}_{\mathsf{O}'}(A) \leq \max_A \mathbf{Adv}^{\text{obl}}_{\mathsf{O}}(A)$. This follows because any adversary against O' can ignore all accesses before the final access and have the same advantage against O.

Since O' is one-round, $p(s + \ell \log N_1) < C \cdot N_1$, and $1 \leq p < C\sqrt{N_1}$ this contradicts Theorem 3. □

7 Conclusion and Open Problems

Lower bounds for ORAM schemes have been largely focused on bandwidth cost for ORAM with an unrestricted number of rounds and constant client memory.

However, there are still open questions when schemes are restricted to have a fixed number of rounds.

We prove near-optimal results for one-round ORAM with large client memory in this paper. However, it is possible that do not we have a tight bound for one-round ORAM with constant client memory. It seems likely that one-round ORAM with constant memory should require $\Omega(N_1)$ overhead.

There is the problem of extending our work out of the balls-in-bins model. Our techniques do not immediately give lower bounds in an information theoretic model for ORAM, but possibly could be extended with techniques similar to those used by Larsen and Nielsen [18]. Many of the proof steps extend to equivalent statements with compression arguments, however it is unclear how to extend Eq. (8) to the information theoretic setting.

This issue is related to an issue which arose with bounded duplicate ORAM. If we bound the duplication, the proof extends but gets weakens significantly. We are unaware of any duplicate balls-in-bins ORAM constructions that match our bound, and it seems likely the loss to duplicates is an artifact of the proof.

Extension beyond partition-restricted to two-round or even an arbitrary k-round is still open. One might hope that the k-round construction from Sect. 6 is tight up to poly-log factors and that the true lower bound for k-round is $\Omega(kN_1^{1/k})$ with constant client memory.

Acknowledgements. We thank the anonymous referees for many suggestions on improving the presentation of this paper. The first and third authors were supported in part by NSF CNS-1928767.

References

1. Asharov, G., Komargodski, I., Lin, W.-K., Nayak, K., Peserico, E., Shi, E.: OptORAMa: optimal oblivious RAM. In: Canteaut, A., Ishai, Y. (eds.) EUROCRYPT 2020. LNCS, vol. 12106, pp. 403–432. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-45724-2_14
2. Boyle, E., Naor, M.: Is there an oblivious RAM lower bound? In: Sudan, M., (ed.) ITCS 2016: 7th Conference on Innovations in Theoretical Computer Science, pp. 357–368, Association for Computing Machinery, Cambridge, 14–16 January 2016
3. Chan, T.-H.H., Chung, K.-M., Shi, E.: On the depth of oblivious parallel RAM. In: Takagi, T., Peyrin, T. (eds.) ASIACRYPT 2017. LNCS, vol. 10624, pp. 567–597. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-70694-8_20
4. Dautrich Jr., J.L., Stefanov, E., Shi, E.: Burst ORAM: minimizing ORAM response times for bursty access patterns. In: Fu, K., Jung, J. (eds.) USENIX Security 2014: 23rd USENIX Security Symposium, pp. 749–764, USENIX Association, San Diego, 20–22 August 2014
5. Devadas, S., van Dijk, M., Fletcher, C.W., Ren, L., Shi, E., Wichs, D.: Onion ORAM: a constant bandwidth blowup oblivious RAM. In: Kushilevitz, E., Malkin, T. (eds.) TCC 2016. LNCS, vol. 9563, pp. 145–174. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-49099-0_6
6. Fletcher, C., Naveed, M., Ren, L., Shi, E., Stefanov, E.: Bucket ORAM: single online roundtrip, constant bandwidth oblivious RAM. Cryptology ePrint Archive, Report 2015/1065 (2015). http://eprint.iacr.org/2015/1065

7. Garg, S., Lu, S., Ostrovsky, R.: Black-box garbled RAM. Cryptology ePrint Archive, Report 2015/307 (2015). http://eprint.iacr.org/2015/307
8. Garg, S., Lu, S., Ostrovsky, R., Scafuro, A.: Garbled RAM from one-way functions. In: Servedio, R.A., Rubinfeld, R. (eds.) 47th Annual ACM Symposium on Theory of Computing, pp. 449–458. ACM Press. Portland, 14–17 June 2015
9. Garg, S., Mohassel, P., Papamanthou, C.: TWORAM: Efficient oblivious RAM in two rounds with applications to searchable encryption. In: Robshaw, M., Katz, J. (eds.) Advances in Cryptology, CRYPTO 2016, Part III, LNCS, vol. 9816, pp. 563–592, Santa Barbara, 14–18 August 2016. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-53015-3_20
10. Gentry, C., Halevi, S., Lu, S., Ostrovsky, R., Raykova, M., Wichs, D.: Garbled RAM revisited. In: Nguyen, P.Q., Oswald, E. (eds.) EUROCRYPT 2014. LNCS, vol. 8441, pp. 405–422. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-642-55220-5_23
11. Goldreich, O., Ostrovsky, R.: Software protection and simulation on oblivious RAMs. J. ACM **43**(3), 431–473 (1996)
12. Goodrich, M.T., Mitzenmacher, M.: Privacy-preserving access of outsourced data via oblivious RAM simulation. In: Aceto, L., Henzinger, M., Sgall, J. (eds.) ICALP 2011. LNCS, vol. 6756, pp. 576–587. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-22012-8_46
13. Goodrich, M.T., Mitzenmacher, M., Ohrimenko, O., Tamassia, R.: Practical oblivious storage. In: Proceedings of the Second ACM Conference on Data and Application Security and Privacy, CODASPY 2012, pp. 13–24, Association for Computing Machinery, New York (2012)
14. Goodrich, M.T., Mitzenmacher, M., Ohrimenko, O., Tamassia, R.: Privacy-preserving group data access via stateless oblivious RAM simulation. In: Rabani, Y. (ed.) 23rd Annual ACM-SIAM Symposium on Discrete Algorithms, ACM-SIAM, pp. 157–167, Kyoto, 17–19 January 2012
15. Hubáček, P., Koucký, M., Král, K., Slívová, V.: Stronger lower bounds for online ORAM. In: Hofheinz, D., Rosen, A. (eds.) TCC 2019. LNCS, vol. 11892, pp. 264–284. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-36033-7_10
16. Jacob, R., Larsen, K.G., Nielsen, J.B.: Lower bounds for oblivious data structures. In: Chan, T.M. (ed.) 30th Annual ACM-SIAM Symposium on Discrete Algorithms, ACM-SIAM, pp. 2439–2447, San Diego, 6–9 January 2019
17. Kushilevitz, E., Lu, S., Ostrovsky, R.: On the (in)security of hash-based oblivious RAM and a new balancing scheme. In: Rabani, Y. (ed.) 23rd Annual ACM-SIAM Symposium on Discrete Algorithms, ACM-SIAM, pp. 143–156, Kyoto, 17–19 January 2012
18. Larsen, K.G., Nielsen, J.B.: Yes, there is an oblivious RAM lower bound!. In: Shacham, H., Boldyreva, A. (eds.) CRYPTO 2018. LNCS, vol. 10992, pp. 523–542. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-96881-0_18
19. Lu, S., Ostrovsky, R.: Distributed oblivious RAM for secure two-party computation. In: Sahai, A. (ed.) TCC 2013. LNCS, vol. 7785, pp. 377–396. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-36594-2_22
20. Lu, S., Ostrovsky, R.: How to garble RAM programs? In: Johansson, T., Nguyen, P.Q. (eds.) EUROCRYPT 2013. LNCS, vol. 7881, pp. 719–734. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-38348-9_42
21. Moataz, T., Mayberry, T., Blass, E.-O.: Constant communication ORAM with small blocksize. In: Ray, I., Li, N., Kruegel, C. (eds.) ACM CCS 2015: 22nd Conference on Computer and Communications Security, pp. 862–873. ACM Press, Denver, 12–16 October 2015

22. Patel, S., Persiano, G., Raykova, M., Yeo, K.: PanORAMa: oblivious RAM with logarithmic overhead. In: Thorup, M. (ed.) 59th Annual Symposium on Foundations of Computer Science, pp. 871–882, IEEE Computer Society Press, Paris, 7–9 October 2018
23. Shi, E., Chan, T.-H.H., Stefanov, E., Li, M.: Oblivious RAM with $O((\log N)^3)$ worst-case cost. In: Lee, D.H., Wang, X., (eds.) Advances in Cryptology - ASIACRYPT 2011. LNCS, vol. 7073, pp. 197–214, Seoul, South, Springer, Heidelberg, 4–8 December 2011. https://doi.org/10.1007/978-3-642-25385-0_11
24. Stefanov, E., et al.: Path ORAM: an extremely simple oblivious RAM protocol. In: Sadeghi, A.-R., Gligor, V.D., Yung, M., (eds.) ACM CCS 2013: 20th Conference on Computer and Communications Security, pp. 299–310. ACM Press, Berlin, 4–8 November 2013
25. Wang, X., Chan, T.-H.H., Shi, E.: Circuit ORAM: on tightness of the Goldreich-Ostrovsky lower bound. In: Ray, I., Li, N., Kruegel, C. (eds.) ACM CCS 2015: 22nd Conference on Computer and Communications Security, pp. 850–861. ACM Press, Denver 12–16 October 2015
26. Williams, P., Sion, R., Single round access privacy on outsourced storage. In: Yu, T., Danezis, G., Gligor, V.D. (eds.) ACM CCS 2012: 19th Conference on Computer and Communications Security, pp. 293–304. ACM Press, Raleigh, 16–18 October 2012