Length Leakage in Oblivious Data Access Mechanisms

Grace Jia
Yale University

Rachit Agarwal
Cornell University

Anurag Khandelwal
Yale University

Abstract

This paper explores the problem of preventing length leakage in oblivious data access mechanisms with passive persistent adversaries. We show that designing mechanisms that prevent both length leakage and access pattern leakage requires navigating a three-way tradeoff between storage footprint, bandwidth footprint, and the information leaked to the adversary. We establish powerful lower bounds on achievable storage and bandwidth footprints for a variety of leakage profiles, and present constructions that perfectly or near-perfectly match the lower bounds.

1 Introduction

Cloud services offer scalable, fault-tolerant, and easy-to-manage systems for storing and querying data. However, using (untrusted) cloud services also leads to significant security concerns: data accesses that used to be contained within an organization's trusted domain are now visible to potentially untrusted entities on the cloud. A now-long line of work has shown that, even if offloaded data is encrypted, an adversary can exploit data access patterns to learn damaging information about the data [1–4]. To protect against such access pattern attacks, our community has developed a large and active body of research on oblivious data access mechanisms: on different adversarial settings (*e.g.*, active [5] vs. persistent passive [6]), enabling transactional [7] and asynchronous queries [8, 9], enabling distributed proxy deployments [10, 11], achieving performance scalability [12–19], and identifying performance limits [20–25], to name a few.

This paper explores a complementary problem: preventing length leakage in oblivious data access mechanisms. Specifically, both ORAM-based [5] and PANCAKE-based [6] oblivious data access mechanisms assume that all data objects are of the same size; this is far from realistic in practice—recent studies from real-world production systems have established that data object sizes can vary by multiple orders of magnitude [26]. For such realistic scenarios, it becomes important to design systems that not only enable oblivious data access but also protect against length leakage.

Preventing length leakage in oblivious data access mechanisms raises new challenges: as we will show, it requires navigating a complex three-way tradeoff between the storage used by the mechanism (i.e., its storage footprint), the total amount of access traffic to the storage per query (i.e., its bandwidth footprint), and the precise information leaked to the adversary. To provide some intuition, consider the following two extreme design points. One obvious design is "padding"-based [27]: each object is padded with random bits to the largest object size, say, s_{max} (queries are responded to in an obvious manner); with n objects, this design requires storage footprint equal to $n \cdot s_{max}$, bandwidth footprint proportional to s_{max} per query[1], and informally speaking, leaks information about the number of objects (n) and the largest object size (s_{max}) to the adversary. At the other extreme are more recent "packing"-based designs [28], in particular size-locked indexes [29]: all data objects are packed into one blob (each data access query is responded to by downloading the entire blob). This requires storage and bandwidth footprint equal to the sum of the sizes of the plaintext objects—$\sum_i s_i$; informally, such packing-based designs leak information about the sum of the sizes of the plaintext objects to the adversary. Returning to the three-way tradeoff: the first design has *higher* storage footprint, *lower* bandwidth footprint, and (informally) *less* information leakage than the second design. Thus, depending on the storage budget, bandwidth budget, and/or security goals, either of the two design points may be of interest.

This paper explores fundamental limits *and* achievable design points for preventing length leakage in oblivious data access mechanisms. We focus on passive persistent adversaries, as motivated by the PANCAKE work [6]. We provide details on our system and security model in §2, but briefly, an encryption proxy takes as input a collection of n plaintext objects of sizes $S = \{s_1, s_2, \ldots, s_{max}\}$, and creates encrypted objects that will be stored in an untrusted storage server. The

[1] ORAM and PANCAKE incur $\Omega(\lg(n)) \cdot s_{max}$ and $3 \cdot s_{max}$ bandwidth footprint, respectively.

adversary can observe all the (encrypted) queries/responses between the proxy and storage, but it cannot inject queries; instead, queries are sampled from a distribution π. The encryption mechanism has an estimate of the distribution, and the adversary knows the precise distribution. The adversary "wins" if it can distinguish the sequence of queries from a sequence of uniformly distributed accesses to fixed-sized random bit strings—capturing both access pattern and length leakage. We introduce a formal security framework that captures this setting as real-or-random indistinguishability under chosen distribution and length attack (ROR-CDLA, §7).

1.1 Summary of Key Results

We establish the following results (Table 1):

(1) First, we establish the intuitive result that padding-based schemes [27] applied to oblivious data access mechanisms [6] yield bandwidth-optimal constructions, while providing strong security: leaking only the number of objects (n) and the largest object size (s_{max}) to the adversary. In particular, we establish that any scheme that reveals only n and s_{max} must incur a minimum bandwidth footprint $\alpha \times s_{max}$ per query, where $\alpha \geq 1$ is a constant (independent of n) for any fixed storage footprint; larger storage footprint may enable a smaller α. Furthermore, we establish that padding-based schemes [27] applied to oblivious data access mechanisms [6] achieve an upper bound that precisely matches this lower bound. These results imply that it is possible to design mechanisms that offer strong security at low bandwidth overheads[2] if there is little variance in object sizes (i.e., the average object size is close to the largest) but quite high for use cases where object sizes may vary by orders of magnitude [26]. This motivates the relaxation of security for more storage and bandwidth-efficient schemes.

(2) Second, we establish lower and upper bounds for packing-based designs [28, 29]. Specifically, our lower bound shows that when restricted to minimum possible storage—the sum of all the object sizes—any scheme that reveals n and $\sum_i s_i$ must have a bandwidth overhead of $\sum_i s_i$ (matching the bandwidth overhead of the size-locked index construction [29]). Our lower bound and the construction whose performance meets the lower bound is more general: given even slightly more storage than $\sum_i s_i$, it is possible to dramatically reduce bandwidth footprint, albeit with slightly more leakage than size-locked indexes. In contrast to size-locked indexes that pack all objects into one bin, our generalized construction first bin-packs objects across s_{max}-sized bins and then employs a PANCAKE-based oblivious access mechanism over the bins. This construction achieves our generalized lower-bounds, i.e., if a scheme is allowed to leak the plaintext object sizes (instead of the sum of the object sizes as in the size-locked index construction), its bandwidth footprint lower bound decreases to a smaller multiple of s_{max} (compared to (1)), with the multiple decreasing even faster as the given storage increases. This is particularly useful for datasets with objects of very variable sizes, where bin-packing objects can permit massive reduction of the bandwidth overhead for a given storage footprint.

Leaked information	Bandwidth Lower-bound	Optimal Scheme
n, s_{max} (§3)	$\frac{\lfloor \kappa n \rfloor}{\lfloor \kappa n \rfloor - n + 1} \cdot s_{max}$	Padded PANCAKE
n, S (§4)	$\frac{\lfloor \kappa n \rfloor}{\lfloor \kappa n \rfloor - N_{OPT} + 1} \cdot s_{max}$[3]	Stuffed PANCAKE
n, s_{max}, π (§5)	No closed form	Greedy PANCAKE
n, S, π (§6)	No closed form	MILP

Table 1: **Summary of our results** for n objects with sizes $S = \{s_1, \ldots, s_n\}$ and access distribution π. κ is a parameter that bounds storage to $\kappa \cdot n \cdot s_{max}$, and N_{OPT} is the minimum number of s_{max}-sized bins required to contain all objects.

(3) Third, for the case of leaking the access distribution, we present an improvement over the existing state-of-the-art construction, PANCAKE [6]. Our scheme greedily replicates objects with higher access probabilities and is provably bandwidth-optimal. Since the scheme can leak the access distribution, the bandwidth footprint depends on the distribution and can often be much smaller than those for (1). This is particularly useful for real-world key-value stores where accesses follow well-known Zipfian or bimodal distributions [26, 32].

(4) Finally, for the case of leaking both the individual object sizes and the access distribution, we demonstrate a fundamental limitation in existing constructions in terms of achieving bandwidth optimality. We demonstrate that existing oblivious data access mechanisms perform *uniform* load balancing of requests across object replicas; however, non-uniform load balancing is necessary to achieve optimality. We present a Mixed-Integer Linear Program (MILP)-based construction that performs non-uniform load-balancing to achieve bandwidth-optimality, outperforming all PANCAKE-variants introduced above. Depending on the input object sizes and access distribution across them, this construction can achieve a significantly low bandwidth overhead. However, the MILP approach does not scale beyond a few hundred objects, so we approximate its behavior using a polynomial-time scheme (Greedy Stuffed PANCAKE, §6.2).

In §2.3, we discuss how an adversary can use each leakage profile to extract meaningful information about the data.

Practical implications. We also empirically characterize the three-way tradeoff space for real-world datasets. Figure 1 shows the bandwidth overhead of each of the constructions in Table 1 as a function of their storage overhead.

Figure 1 confirms our results (1)-(4): bandwidth overheads for each scheme decrease with more storage, and given any

[2] A scheme's bandwidth overhead is its bandwidth footprint relative to an insecure baseline's footprint. A similar definition holds for storage overhead.

[3] Bound only holds for $\hat{s} = s_{max}$ and $\hat{s} \geq \frac{\lfloor \kappa n \rfloor}{\lfloor \kappa n \rfloor - N_{OPT} + 1} \cdot s_{max}$.

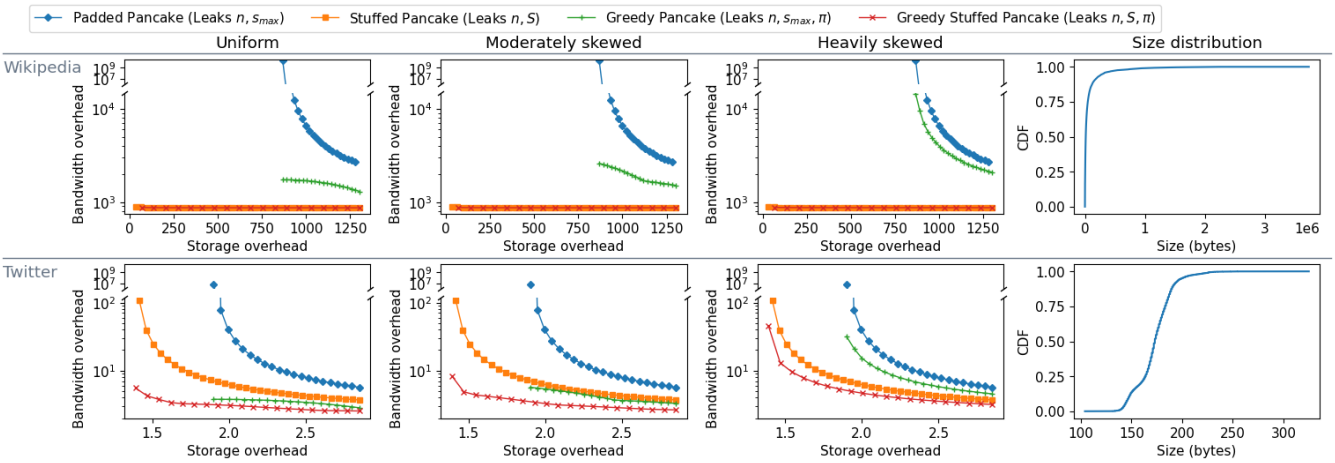


Figure 1: **Bandwidth vs. storage overhead across schemes with different leakage (§1.1).** We use real-world datasets from Twitter [30] (top) and Wikipedia [31] (bottom) (see Appendix for Facebook datasets [26]) and Zipf access patterns with varying skew (left to right). The rightmost column shows the dataset's object size CDF. Bandwidth overheads for schemes that do not leak object sizes start from $n \cdot s_{max}$ (required for security). The y-axes are in log-scale broken between bandwidth overheads 120–$10^6\times$ (Twitter) and 15000–$10^6\times$ (Wikipedia).

fixed storage, bandwidth overheads decrease as more information is leaked. Considering dataset-specific characteristics, we find that since Wikipedia has a large range of object sizes, all schemes observe a minimum of $\sim$1000$\times$ overhead, since hiding each object's identity requires each query to fetch the largest object size, which is 1000$\times$ larger than the average (similar observations have been made in prior work [33]). However, schemes that leak the object lengths (as noted in (2), (4)) incur lower bandwidth overheads, confirming our theoretical results. As such, the bandwidth overhead for such schemes is over 10$\times$ lower than schemes that do not leak object sizes. On the other hand, the Twitter dataset with less variability in object sizes can use schemes that do not leak object sizes for stronger security at a lower bandwidth overhead. Specifically, it observes $<$ 10$\times$ bandwidth overhead with 2–3$\times$ storage overhead, which reduces to 2–3$\times$ bandwidth overhead on leaking object sizes.

For access patterns with varying skew, schemes that do not leak the access distribution have the same bandwidth overhead regardless of skew, as expected; for schemes that leak access distribution, we observe lower bandwidth overheads (as noted in (3), (4)). Bandwidth overheads increase with the access skew since hiding the identity of the most popular object requires injecting more noise for all other objects.

Takeaways. For the problem of simultaneously preventing length leakage and access pattern leakage, our results establish a rich tradeoff space between the storage footprint, bandwidth footprint, and information leakage. Our constructions demonstrate that existing oblivious data access mechanisms can resist length leakage attacks with simple modifications in proxy-side encryption mechanisms. Even so, this paper only takes the first step in laying the intellectual foundation for preventing length leakage in oblivious data access mechanisms. Much more work needs to be done in this space; to that end, we close the paper by outlining several avenues of future research.

2 Overview

We now describe our system and security models.

2.1 System Model

Our setup consists of a key-value (KV) store that supports (single-key) get and put operations on KV pairs (k, v) submitted by one or more clients. Our results can, however, be applied to any data store that supports read and write operations. We focus on a trusted proxy architecture commonly used by encrypted data stores [8, 10–12, 34], which assumes multiple client applications route query requests through a single trusted proxy. The proxy manages the execution of these queries on behalf of the clients, sending queries to the untrusted storage service. We assume all communication channels are encrypted, e.g., using TLS.

2.2 Security Model

We now define our adversary and class of considered schemes.

Threat model. We use a trusted proxy threat model where the client and proxy servers belong to a trusted domain. We model client queries as independent samples from a distribution π over keys, i.e., the probability of access to a key k is $\pi(k)$. This work focuses on settings where the underlying distribution π is static. We discuss how our results may translate to the dynamic setting where π changes over time in §8.

We assume that the untrusted cloud storage service is controlled by a passive persistent adversary. This is typical of secure cloud-deployments [1, 2, 4, 6, 35], where the adversary (usually the cloud provider) can observe the volume (i.e., the number and size) of client queries, as well as the encrypted data stored at and retrieved from the storage service. It cannot, however, access cryptographic keys or change queries, responses, or stored data. The adversary aims to infer any information about individual KV pairs that are accessed. Similar to prior work [6, 9, 10, 12], we do not target hiding the timing of client queries.

Considered schemes. We consider a class of schemes $\mathcal{M}$, where each scheme $\Pi \in \mathcal{M}$ is defined as a pair of algorithms $\Pi = (\mathsf{Init}, \mathsf{Query})$. While we defer the formal security definitions to §7, we focus on an intuitive description here.

Init is an initialization algorithm that takes as input:

- a set KV of n key-value pairs $\{(k_1, v_1), \ldots, (k_n, v_n)\}$,
- a set S of value sizes $\{s_1, \ldots, s_n\}$, where s_i is the size of v_i.
- a distribution π over KV, where $\pi(k_i)$ corresponds to the probability of accessing key k_i, and,
- a storage parameter σ, which decides the total storage capacity available to the scheme.

Init outputs the following:

- a new set EKV of $\hat{n}$ KV pairs $\{(\hat{k}_1, \hat{v}_1), \ldots, (\hat{k}_{\hat{n}}, \hat{v}_{\hat{n}})\}$,
- a mapping P that maps each key k_i in KV to a set of keys in EKV whose values "contain" v_i in a contiguous sequence of s_i bytes. If Keys(KV) and Values(KV) denote the set of keys and values in KV respectively, then:

$$\forall k \in \mathsf{Keys}(\mathsf{KV}), \mathsf{P}(k) = \{\hat{k} \in \mathsf{Keys}(\mathsf{EKV}) \mid \hat{v} \text{ contains } v\}$$

- a set π_r of functions indexed by key k, where $\pi_r[k]$ takes as input $\hat{k} \in \mathsf{Keys}(\mathsf{EKV})$ and outputs the access probability of the copy of k's value contained in $\hat{k}$'s encrypted value [4],
- a distribution π_f over EKV, where $\pi_f(\hat{k})$ denotes the probability of accessing key $\hat{k}$ via "fake" queries or noise,
- the proportion of real queries δ to EKV, and,
- a constant $B \geq 1$ that will be used by Query and determine bandwidth footprint.

The mapping P is stored at the proxy and is hidden from the adversary; so are π_r and π_f. Note that since values of KV are contained in a contiguous sequence of s_i bytes in values of EKV (identified by P), $\mathcal{M}$ implicitly rules out compression or other encoding schemes that increase or decrease the size of data contained in values $v \in \mathsf{Values}(\mathsf{KV})$. Each key-value pair $(\hat{k}, \hat{v})$ in EKV is ultimately encrypted by applying a secretly keyed pseudorandom function (PRF) to the key (*e.g.*, HMAC) to generate a label, denoted $F(\hat{k})$, and symmetrically encrypting the value using authenticated encryption, denoted $E(v')$. These labels and encrypted values are ultimately stored in the storage service, while the secret keys needed for F and E are stored at the proxy and, therefore, hidden from an adversary. Because F is deterministic, the proxy can perform operations for key $\hat{k}$ by instead requesting $F(\hat{k})$[5]. We have omitted the two required cryptographic secret keys in our notation for simplicity. We cryptographically bind labels and value ciphertexts by using the label as associated data with E.

EKV additionally satisfies the following three properties:

Property 1. *Each value $\hat{v}$ in* Values(EKV) *has the same size* $\hat{s} \geq \max_i(s_i)$.

Property 2. *The total storage used by values* Values(EKV) *must be less than σ, i.e., $\hat{s} \times \hat{n} \leq \sigma$.*

Property 3. *Each v_i in* Values(KV) *is "contained" in at least one $\hat{v}_j$ in* Values(EKV)*, i.e., $\forall k_i \in \mathsf{Keys}(\mathsf{KV}), |\mathsf{P}(k_i)| \geq 1$.*

Intuitively, Property 1 ensures protection against length-leakage, while Property 2 bounds the scheme's storage footprint. Finally, Property 3 ensures that all the data in KV is still present in EKV, i.e., the scheme is lossless.

Query is a query algorithm that transforms a query on plaintext keys in Keys(KV) to one or more queries on encrypted keys in Keys(EKV). Query takes as input:

- Init's outputs P, π_r, π_f, δ, B and,
- a sequence of queries $Q = \{\mathsf{q}_1, \ldots, \mathsf{q}_q\}$ on plaintext keys in Keys(KV), where each query q_j is drawn from the distribution π; i.e., $\Pr[\mathsf{q}_j = k_i] = \pi(k_i)$.

Query generates as output:

- a sequence of queries $\hat{Q} = \{\hat{\mathsf{q}}_1, \ldots, \hat{\mathsf{q}}_{\hat{q}}\}$ over encrypted keys in Keys(EKV), where $\hat{q} = q \cdot B$.

The generated sequence of queries satisfies two properties:

Property 4. *All keys in* EKV *are accessed with equal probability, i.e., $\forall i, j,\ \Pr[\hat{\mathsf{q}}_j = F(\hat{k}_i)] = 1/\hat{n}$.*

Property 5. *The expected number of accesses to any key $k \in$* Keys(KV) *is no more than the expected number of accesses to all keys $\hat{k} \in$* Keys(EKV) *that k maps to in* P*:*

$$\forall k \in \mathsf{Keys}(\mathsf{KV}), E(k \in Q) \leq \sum_{\hat{k} \in \mathsf{P}(k)} E(\hat{k} \in \hat{Q}).$$

Property 4 ensures security by requiring protection against access pattern leakage. Property 5 ensures correctness by requiring that all queries in Q are answered in expectation over the randomness in Query's execution. We use "in expectation"

[4] Note that the function $\pi_r[k]$ is *not* a distribution over $\mathsf{P}(k)$, since $\sum_{\hat{k}} \pi_r[k](\hat{k}) = \pi(k)$, i.e., the original access probability of key k and not 1.

[5] There is a negligible probability for two distinct keys $\hat{k}_i, \hat{k}_j$ to have $F(\hat{k}_i) = F(\hat{k}_j)$; we ignore this case for simplicity.

rather than "with high probability" to strengthen our results: since our lower bounds (§3, §4) hold for the more relaxed expectation-based correctness, they must also hold for more stringent probability-based correctness. Property 5 thus excludes from $\mathcal{M}$ any mechanism that achieves security (i.e., Properties 1 and 4) "incorrectly", *e.g.*, by ignoring a large fraction of client queries or responding with incorrect values.

We base our analysis on two key performance metrics:

- The **storage footprint** of Π is defined as σ, the total storage capacity available to the scheme.
- The **bandwidth footprint** of Π is defined as $\beta := B \cdot \hat{s}$, the number of bytes fetched from EKV per query in Q.

2.3 Considered Leakage Profiles

How much information can be leaked about the underlying data store differs by application; we capture these standards in the leakage profile $\mathcal{L}$ atop our considered class of schemes $\mathcal{M}$. Leakage profiles describe an upper bound on what an adversary learns from the execution of some scheme $\Pi \in \mathcal{M}$: any information other than the output of $\mathcal{L}$ must be hidden.

Again, we only provide an intuitive description of the security goal with leakage here and defer a formal simulator-based ROR-CDLA security definition to §7. Given any input store KV with access distribution π, value sizes S, storage footprint σ, and a sequence Q of q queries sampled from π, we consider the "real world" where Π produces a sequence of (encrypted) queries $\hat{Q}$. In contrast, the "ideal world" uses a sequence of $q \cdot B$ accesses generated from a uniform distribution over a set of $\hat{n}$ random bit strings of size $\hat{s}$. For some leakage profile $\mathcal{L}$, a scheme $\Pi \in \mathcal{M}$ is $\mathcal{L}$-secure (or, $\Pi \in \mathcal{ML}$) if the outputs of the real and ideal worlds are indistinguishable. In the ideal world, $\hat{n}$, $\hat{s}$ and B are generated by an algorithm $\mathcal{S}$ that only sees $\mathcal{L}(\mathsf{KV}, S, \pi, \sigma)$. In other words, any adversary's view of Π's execution must be simulatable given only $\mathcal{L}$'s output and no other information about Π's inputs or internal state. In §7, we will show that the real and ideal worlds correspond to ROR-CDLA^0 and ROR-CDLA^1, respectively.

We emphasize that these leakage profiles only relax what information is revealed during the *initialization* of the encrypted store EKV. Any $\Pi \in \mathcal{M}$ must still meet Properties 1-5, ensuring that *for every query, the identity of the accessed key and its associated value are hidden*. Thus, all schemes in our considered class $\mathcal{M}$ resist existing access pattern attacks [1, 36, 37] and length leakage attacks [2, 38, 39], since they rely on leakage due to accesses to the KV store via client queries, i.e., information revealed *after* initialization.

We focus on four leakage profiles that determine whether or not the scheme reveals KV's value sizes or access distribution:

$\mathcal{L}_m$ (§3): $\mathcal{ML}_m$ is the subset of schemes in $\mathcal{M}$ with leakage profile $\mathcal{L}_m$, where $\mathcal{L}_m(\mathsf{KV}, S, \pi, \sigma) = (n, s_{max}, B)$. Intuitively, an adversary only learns the number of keys and the maximum value size in KV, along with the number of encrypted queries Π makes, $q \cdot B$, ensuring protection against both access distribution and length leakage-abuse attacks. Since our considered leakage profiles have the same input and outputs that always contain n, B, we will omit them in our notation.

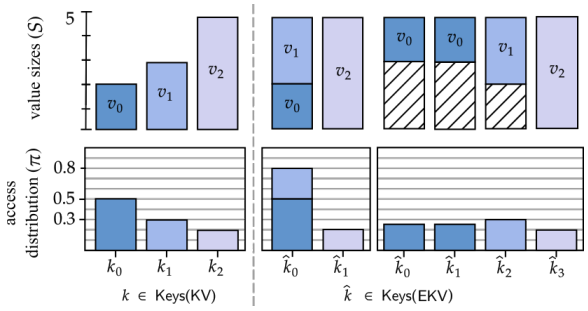
$\mathcal{L}_S$ (§4): $\mathcal{ML}_S$ denotes the class of schemes in $\mathcal{M}$ with leakage profile $\mathcal{L}_S$, where $\mathcal{L}_S$ outputs (S). For any scheme $\Pi \in \mathcal{ML}_S$, an adversary can observe the sizes of all values in KV S and the output of $\mathcal{L}_m$. This permits, for instance, compression-based attacks in the vein of CRIME [38] and BREACH [39], which leverage the sizes of encrypted values to learn about the plaintext values. As a concrete example, consider two sensitive database tables with noticeably different value size distributions, *e.g.*, flu records in one table (with smaller, less detailed reports) and cancer records in another (with longer, more detailed reports), in a medical database. In this setting, the attacker can distinguish between the tables and identify when a request is for a flu or cancer record. This is analogous to attacks in traffic analysis literature over websites, which distinguish between different websites by observing the sum of packet sizes across its visited webpages [40].

$\mathcal{L}_\pi$ (§5): $\mathcal{ML}_\pi$ denotes the class of schemes in $\mathcal{M}$ with leakage profile $\mathcal{L}_\pi$, where $\mathcal{L}_\pi$ outputs (s_{max}, π). For any scheme $\Pi \in \mathcal{ML}_\pi$, an adversary can identify the access distribution π over KV and the output of $\mathcal{L}_m$. Like in the previous setting, an attacker can distinguish between two database tables by their access distributions, *e.g.*, a table with flu records and uniform access pattern from a table with cancer records and skewed access pattern. Note that $\mathcal{L}_\pi$ still excludes traditional access pattern attacks [1, 2, 36, 37] that exploit the mapping between KV pairs and their access frequencies.

$\mathcal{L}_{S\pi}$ (§6): $\mathcal{ML}_{S\pi}$ denotes the class of schemes in $\mathcal{M}$ with leakage profile $\mathcal{L}_{S\pi}$, where $\mathcal{L}_{S\pi}$ outputs (S, π). For any scheme $\Pi \in \mathcal{ML}_{S\pi}$, an adversary can identify the value lengths S *and* access distribution π. In this setting, attacks on schemes can exploit the leakage of size distribution (i.e., attacks in $\mathcal{L}_S$), access distribution (i.e., attacks in $\mathcal{L}_\pi$), or a combination of the two (*e.g.*, a volume *and* access pattern leakage-abuse attack [41]). Continuing our medical database example, a third table of arthritis records with uniform access patterns and longer reports can be distinguished from the cancer table by value sizes and the flu table by access pattern.

3 $\mathcal{L}_m$: Revealing the largest value size

Under our strictest leakage profile, the adversary may only learn the largest value size (s_{max}) and the number of keys (n). We do not explore stricter profiles that hide even s_{max} since they would yield impractically high bandwidth footprint overheads (even $\mathcal{L}_m$ yields impractical overheads as noted in §1.1). To understand what constructions are possible under this leakage profile, consider a KV with three key-value pairs, with sizes and access probabilities, as shown in Figure 2a.



(a) Example input KV. (b) Packing. (c) Padding + replication.
Figure 2: Sample constructions under $\mathcal{M}$; see §3 for details.

First, since no value size other than $s_{max} = 5$ can be leaked, all values in the output EKV must be at least 5 bytes in size.

It may be tempting to pack the values of keys k_0 and k_1 into a single 5-byte value for space efficiency (Figure 2b); however, this scheme is $\notin \mathcal{ML}_m$. By observing that EKV consumes 10 bytes of storage, the adversary learns a bound on the sum of the value lengths and that at least one value is smaller than s_{max}. As such, for any scheme in $\mathcal{ML}_m$, its storage footprint must be independent of the value sizes S. We, therefore, focus on schemes that pad the values for k_0 and k_1 to size s_{max}.

The second security requirement is a uniform access distribution across keys in EKV. Consider a scheme that injects "fake" accesses to each padded value to equalize the access probabilities of the keys. However, the additional volume of access traffic to EKV reveals information about the original access distribution over the input KV, *e.g.*, an access distribution skewed towards k_0 in Figure 2a requires more fake accesses to k_1 and k_2, while one where all key access probabilities are equal does not require any additional fake accesses. An adversary can distinguish between these two cases by observing the traffic volume to EKV. Thus, the bandwidth footprint of any scheme in $\mathcal{ML}_m$ must be independent of the access distribution π over KV.

Other schemes that naively replicate the key-value pairs or combine replication and fake accesses can still be insecure under $\mathcal{ML}_m$. Consider a scheme that replicates the padded value of k_0 and divides its accesses equally between the copy and the original. This results in four padded values (Figure 2c), and the remaining non-uniformity in access distribution can be smoothed using fake accesses. This scheme can still reveal information about the original access distribution over the input KV: if replication is used to smooth access frequencies based purely on the skew in the input distribution, then the total number of replicas (which is output by Init) leaks this degree of skew. In particular, if π were uniform, no replicas would be needed, and the output EKV would only contain three key-value pairs instead of four replicas in our example. As such, the storage footprint of a scheme in $\mathcal{ML}_m$ must also be independent of the access distribution π.

We formalize these intuitive observations, i.e., a scheme $\Pi \in \mathcal{ML}_m$ must have output storage and bandwidth footprints independent of the value sizes and access distributions over KV, into performance lower bounds for any scheme $\Pi \in \mathcal{ML}_m$ in §3.1. We then show in §3.2 that an adaptation of the PANCAKE algorithm [6] — dubbed Padded PANCAKE or PPC — is a bandwidth-optimal $\mathcal{ML}_m$ scheme.

3.1 Performance Bounds

Any scheme $\Pi \in \mathcal{ML}_m$ must output a EKV that has the same storage footprint $(\hat{n} \cdot \hat{s})$ across all input stores KV of n key-value pairs — otherwise it reveals information about the value sizes S, or at the very least, some information about $\sum_i s_i$. For correctness, all data encoded in KV must also be in EKV (Property 3). An input where all values have size s_{max} requires EKV to store at least $n \cdot s_{max}$ bytes for no information to be lost from KV. Then for any n-key input to Π, the output EKV must always have a storage footprint of at least $n \cdot s_{max}$. Moreover, we find that in minimizing bandwidth footprint, schemes in $\mathcal{ML}_m$ do not benefit from using EKV value size $\hat{s}$ larger than s_{max}. This is because larger EKV values increase the number of bytes transferred per access without reducing the proportion of fake queries required. We generalize these observations to all schemes in $\mathcal{M}$ that do not leak S in the following two lemmas, deferring their proofs to Appendix B.

Lemma 1. *Any $\Pi \in \mathcal{M}$ with leakage $\mathcal{L}$ and $S \notin \mathcal{L}$ requires storage footprint $\sigma \geq n \cdot s_{max}$.*

Lemma 2. *Given storage footprint $\sigma = \kappa \cdot n \cdot s_{max}$, a scheme $\Pi \in \mathcal{M}$ with leakage $\mathcal{L}$ and $S \notin \mathcal{L}$ that incurs the minimum possible bandwidth footprint must use $\hat{s} = s_{max}$.*

In Theorem 3.1, we leverage the above lemmas to show that the minimum bandwidth footprint for any scheme in $\mathcal{ML}_m$ with storage footprint $\sigma = \kappa \cdot n \cdot s_{max}$ $(\kappa \geq 1)$ is $\frac{\lfloor \kappa n \rfloor}{\lfloor \kappa n \rfloor - n + 1} \cdot s_{max}$. Since $\mathcal{ML}_m$ schemes do not leak π, their bandwidth footprint must be the same for any π. Intuitively, our proof identifies inputs S and π that maximize any scheme's bandwidth footprint — this happens when all values are as large as possible and access distribution is as skewed as possible. An $\mathcal{ML}_m$ scheme must incur the same bandwidth footprint for any other S and π, so the minimum bandwidth footprint achievable in this worst case is also the lower-bound for any $\mathcal{ML}_m$ scheme.

Theorem 3.1. *Any $\Pi \in \mathcal{ML}_m$ with storage footprint $\sigma = \kappa \cdot n \cdot s_{max}$ $(\kappa \geq 1)$ must have bandwidth footprint $\beta \geq \frac{\lfloor \kappa n \rfloor \cdot s_{max}}{\lfloor \kappa n \rfloor - n + 1}$.*

Proof. Suppose for contradiction that Π incurs storage footprint $\sigma = \kappa \cdot n \cdot s_{max}$ and minimum possible bandwidth footprint $\beta = (1 - \delta) \cdot \frac{\lfloor \kappa n \rfloor}{\lfloor \kappa n \rfloor - n + 1} \cdot s_{max}$ for some $0 < \delta < 1$. We will show that the scheme cannot be $\mathcal{L}_m$-secure.

We define an $\mathcal{L}_m$-adversary $\mathcal{A}$ as in Figure 3. The intuition behind $\mathcal{A}$'s attack — and our proof — is that given a heavily skewed access distribution as input, Π does not have enough

$\mathcal{A}_1$: Computing inputs:

$S \leftarrow \{s_{max}, \ldots, s_{max}\}$
$\pi \leftarrow \{1-\varepsilon, \frac{\varepsilon}{n-1}, \ldots, \frac{\varepsilon}{n-1}\}$
$\sigma \leftarrow \kappa \cdot n \cdot s_{max}$
$\mathsf{KV} \leftarrow \emptyset$
For i in 1 to n:
  $k_i \leftarrow\$ \{0,1\}^m$
  $v_i \leftarrow\$ \{0,1\}^{S[i]}$
  $\mathsf{KV} \leftarrow\cup (k_i, v_i)$
Return $\mathsf{KV}, S, \pi, \sigma$

$\mathcal{A}_2(\mathsf{EKV}, \hat{Q})$: Computing output bit:

Build histogram of accesses in $\hat{Q}$; Let x be the maximum number of accesses to any $\hat{k} \in \mathsf{Keys}(\mathsf{EKV})$.
If $x > (c+3\sqrt{c})\ln\hat{n}$:
  $b \leftarrow 0$
Else:
  $b \leftarrow 1$
Return b

Figure 3: Real-or-random adversary $\mathcal{A}$ for Theorem 3.1. $\varepsilon \to 0$ is chosen by $\mathcal{A}$ to skew the access distribution.

bandwidth to ensure a uniform access distribution over its output EKV (*e.g.*, by injecting fake accesses). We show this in Step 1 of our proof below. This allows the adversary $\mathcal{A}$ to distinguish between ROR-CDLA^0 and ROR-CDLA^1 by examining the sequence of output queries $\hat{Q}$ and performing a simple statistical test: it simply guesses that it is interacting with the real Π if the maximum number of queries to any key in EKV is above some threshold. The threshold must be carefully chosen such that the probability of exceeding it is low for a uniform distribution of queries but high for the non-uniform distribution of queries that Π would generate with insufficient bandwidth. This forms the basis of Step 2, where we show that the adversary has a non-trivial probability of distinguishing between ROR-CDLA^0 and ROR-CDLA^1 for $\delta > 0$, completing the proof.

The attack assumes that $\hat{Q}$ has a sufficiently large number of queries, specifically $\hat{q} = |\hat{Q}| = c\hat{n}\ln\hat{n}$ queries, where $c > 1/\ln\hat{n}$ is a polynomial in $\hat{n}$, ensuring that $\mathcal{A}$'s attack is polynomial-time. Looking ahead, we set the threshold $\mathcal{A}$ uses to be $\hat{q}/\hat{n} + 3\sqrt{c}\ln\hat{n} = (c+3\sqrt{c})\ln\hat{n}$, and $c = \hat{n}^2$.

Step 1: Given $\mathcal{A}$'s choice of inputs, Π's output EKV has a non-uniform access distribution. To this end, we show that $\exists\hat{k} \in \mathsf{Keys}(\mathsf{EKV})$ with access probability $\Pr[\hat{k}] > \frac{1}{\hat{n}}$.

Recall that in the KV generated by $\mathcal{A}_1$, all n values are of size s_{max}, the most heavily accessed key (which we shall call h) has access probability $1-\varepsilon$, and the remaining $n-1$ keys have access probability $\frac{\varepsilon}{n-1}$. Lemma 2 shows that for Π to incur the minimum possible bandwidth it must use $\hat{s} = s_{max}$. Then the number of values in EKV is $\hat{n} = \lfloor\frac{\sigma}{\hat{s}}\rfloor = \lfloor\kappa n\rfloor$, and each value can contain exactly one value from KV. Now for Π to fulfill Property 5, the sum of accesses to each $\hat{h} \in \mathsf{P}(h)$ in $\hat{Q}$ must be at least the number of accesses to h in Q in expectation. This gives the inequality below:

$$(1-\varepsilon)\cdot q \leq \left(\sum_{\hat{h}\in\mathsf{P}(h)} \Pr\left[\hat{h} \in \hat{Q}\right]\right)\cdot q \cdot B$$

$$\text{or, } B \geq \frac{1-\varepsilon}{\sum_{\hat{h}\in\mathsf{P}(h)}\Pr\left[\hat{h}\in\hat{Q}\right]}$$

Since bandwidth footprint is proportional to B, in the best case Π can minimize the bandwidth footprint by splitting the access probability of h uniformly across $\mathsf{P}(h)$, i.e.,

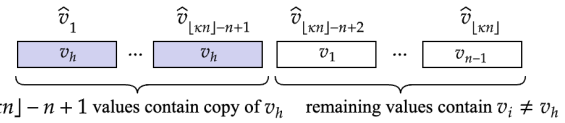


Figure 4: Placement of values $v \in \mathsf{Values}(\mathsf{KV})$ across values $\hat{v} \in \mathsf{Values}(\mathsf{EKV})$ to minimize bandwidth footprint with $\mathcal{L}_m$ leakage. v_h (highlighted) is the value associated with key h in Theorem 3.1.

$$B \geq \frac{1-\varepsilon}{|\mathsf{P}(h)|\cdot p} \tag{1}$$

where p is the probability of accessing any key $\hat{h} \in \mathsf{P}(h)$. Since Π controls $|\mathsf{P}(h)|$, it can minimize B by maximizing $|\mathsf{P}(h)|$. To do so, Π must assign the $n-1$ keys other than h to as few keys in $\mathsf{Keys}(\mathsf{EKV})$ as possible, i.e., $n-1$ keys. This allows h to be mapped to the remaining $\lfloor\kappa n\rfloor - n + 1$ values in EKV, i.e., $|\mathsf{P}(h)| = \lfloor\kappa n\rfloor - n + 1$, as shown in Figure 4. Substituting $|\mathsf{P}(h)|$ in (1), we lower bound p:

$$p \geq \frac{1-\varepsilon}{B(\lfloor\kappa n\rfloor - n + 1)}$$

Since Π's bandwidth footprint β is

$$B\cdot\hat{s} = (1-\delta)\cdot\frac{\lfloor\kappa n\rfloor}{\lfloor\kappa n\rfloor - n + 1}\cdot s_{max},$$

Rearranging gives us $\frac{1}{\lfloor\kappa n\rfloor(1-\delta)} = \frac{1}{B(\lfloor\kappa n\rfloor - n+1)}$, which plugs into the above lower bound on p to give:

$$p \geq \frac{1}{\lfloor\kappa n\rfloor}\cdot\frac{1-\varepsilon}{1-\delta} = \frac{1}{\hat{n}}\cdot\frac{1-\varepsilon}{1-\delta}.$$

Since EKV contains a key with access probability $p \geq \frac{1}{\hat{n}}\cdot\frac{1-\varepsilon}{1-\delta}$, we know that $\mathcal{A}$ can choose a ε such that $p > \frac{1}{\hat{n}}$, resulting in a non-uniform distribution over keys in EKV.

At this point, a computationally unbounded adversary can already distinguish the non-uniform distribution over the real Π's output from the uniform distribution over a simulated output by observing a large enough sequence of Π's output queries ($\hat{Q}$). However, ROR-CDLA security employs a polynomial-time adversary $\mathcal{A}$; so, we define $\mathcal{A}_2$ in Figure 3 to use the simple threshold-based test to detect non-uniformity in $\hat{Q}$'s access distribution over EKV, where $|\hat{Q}|$ is polynomial in $\hat{n}$. We then show that this test gives $\mathcal{A}$ non-negligible advantage against Π for all simulator algorithms $\mathcal{S}$.

Step 2: $\mathcal{A}$ has a non-negligible advantage. To show this, we lower-bound $\mathbf{Adv}^{\text{ror-cdla}}_{\Pi,\mathcal{L}_m,\mathcal{A},\mathcal{S}}(\lambda)$ by a function $f(\hat{n},\varepsilon,\delta)$, and show that with $\varepsilon \to 0$ and sufficiently large values of $\hat{n}$, $f(\hat{n},\varepsilon,\delta)$ approaches 1 for $0 < \delta < 1$ and $\varepsilon \to 0$. Since

$$\mathbf{Adv}^{\text{ror-cdla}}_{\Pi,\mathcal{L}_m,\mathcal{A},\mathcal{S}}(\lambda) = |\Pr[\text{ROR-CDLA}^0_{\Pi,\mathcal{L}_m,\mathcal{A},q}(\lambda) = 1] - \Pr[\text{ROR-CDLA}^1_{\Pi,\mathcal{L}_m,\mathcal{A},\mathcal{S},q}(\lambda) = 1]|,$$

We obtain this function by finding (i) an upper bound on $\Pr[\text{ROR-CDLA}^0_{\Pi,\mathcal{L}_m,\mathcal{A},q}(\lambda) = 1]$, and (ii) a lower bound on $\Pr[\text{ROR-CDLA}^1_{\Pi,\mathcal{L}_m,\mathcal{A},\mathcal{S},q}(\lambda) = 1]$. We also note that since

$\mathcal{A}_2$'s test does not leverage $\hat{n}$, $\hat{s}$ and B generated by the algorithm $\mathcal{S}$, $\mathcal{A}$ works against all possible $\mathcal{S}$.

Step 2(i): We upper-bound $\Pr[\text{ROR-CDLA}^0_{\Pi,\mathcal{L}_m,\mathcal{A},q}(\lambda)=1]$, i.e., the probability that $\mathcal{A}$ guesses incorrectly that the queries $\hat{Q}$ it is given are sampled from a uniform distribution rather than generated by Π.

To do so, we focus on the key $\hat{h} \in \mathsf{P}(h)$ with access probability p. $\mathcal{A}_2$ outputs 1 if, for all keys in EKV, the number of queries they receive is less than or equal to $\mathcal{A}$'s threshold. We can, therefore, use the probability that $\hat{h}$ receives $\leq (c+3\sqrt{c})\ln\hat{n}$ queries as an upper bound for $\Pr[\text{ROR-CDLA}^0_{\Pi,\mathcal{L}_m,\mathcal{A},q}(\lambda)=1]$, and compute the same using the one-sided Chebyshev inequality. While we defer its full derivation to Appendix B.1, we state the upper bound below:

$$\Pr[\text{ROR-CDLA}^0_{\Pi,\mathcal{L}_m,\mathcal{A},q}(\lambda)=1] \leq \frac{(1-\varepsilon)(\hat{n}(1-\delta)-(1-\varepsilon))}{(1-\varepsilon)(\hat{n}(1-\delta)-(1-\varepsilon))+\frac{c\ln\hat{n}\left(\frac{1-\varepsilon}{1-\delta}-1-\frac{3}{\sqrt{c}}\right)^2}{\hat{n}(1-\delta)^2}}. \quad (2)$$

Note that $\mathcal{A}$ must choose a small enough ε such that $a>0$, specifically $\varepsilon < 1-(1+3\sqrt{c})(1-\delta)$. Since we also need $\varepsilon > 0$ for the π chosen by $\mathcal{A}$ to be a valid distribution, the attack can only work with $\delta > \frac{3}{3+\sqrt{c}}$.

Step 2(ii): We lower-bound $\Pr[\text{ROR-CDLA}^1_{\Pi,\mathcal{L}_m,\mathcal{A},\mathcal{S},q}(\lambda)=1]$, i.e., the probability that $\mathcal{A}$ guesses correctly that the queries in $\hat{Q}$ are sampled from a uniform distribution.

Once again, $\mathcal{A}$ outputs 1 if all keys receive more than $(c+3\sqrt{c})\ln\hat{n}$ queries, so we represent the number of queries to a key in EKV with a binomial random variable. We then use the Chernoff bound to compute the upper bound of the probability that *all* keys receive $> (c+3\sqrt{c})\ln\hat{n}$ queries, which we subtract from 1 to obtain the desired lower bound on $\Pr[\text{ROR-CDLA}^1_{\Pi,\mathcal{L}_m,\mathcal{A},\mathcal{S},q}(\lambda)=1]$. Again, while its derivation is deferred to Appendix B.1, we have the lower bound:

$$\Pr[\text{ROR-CDLA}^1_{\Pi,\mathcal{L}_m,\mathcal{A},\mathcal{S},q}(\lambda)=1] \geq 1-\hat{n}^{-2}. \quad (3)$$

Finally, we compute $\mathcal{A}$'s advantage using (2) and (3):

$$\begin{aligned}\mathbf{Adv}^{\text{ror-cdla}}_{\Pi,\mathcal{L}_m,\mathcal{A},\mathcal{S}}(\lambda) &= |\Pr[\text{ROR-CDLA}^0_{\Pi,\mathcal{L}_m,\mathcal{A},q}(\lambda)=1] \\ &\quad - \Pr[\text{ROR-CDLA}^1_{\Pi,\mathcal{L}_m,\mathcal{A},\mathcal{S},q}(\lambda)=1]| \\ &\geq \left|1-\frac{1}{\hat{n}^2}-\frac{(1-\varepsilon)(\hat{n}(1-\delta)-(1-\varepsilon))}{(1-\varepsilon)(\hat{n}(1-\delta)-(1-\varepsilon))+\frac{c\ln\hat{n}\left(\frac{1-\varepsilon}{1-\delta}-1-\frac{3}{\sqrt{c}}\right)^2}{\hat{n}(1-\delta)^2}}\right| \\ &= f(\hat{n},\varepsilon,\delta)\end{aligned}$$

giving us our lower-bound function f on the advantage. We let $\mathcal{A}$ set $\varepsilon \to 0$ and $c=\hat{n}^2$, giving us:

$$f(\hat{n},\delta) = \left|1-\hat{n}^{-2}-\frac{\hat{n}(1-\delta)-1}{\hat{n}(1-\delta)-1+\frac{\hat{n}^2\ln\hat{n}\left(\frac{1}{1-\delta}-1-\frac{3}{\hat{n}}\right)^2}{\hat{n}(1-\delta)^2}}\right|$$

Figure 5 shows that for large $\hat{n}$, the bound on $\delta > \frac{3}{3+\sqrt{c}}$ for which $\mathcal{A}$'s attack works becomes vanishingly small, and

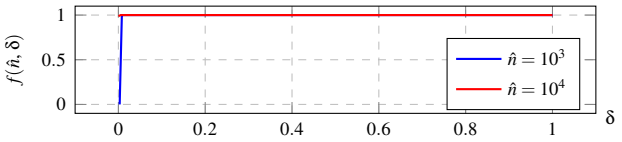


Figure 5: Variation of $f(\hat{n},\delta)$ with δ and $\hat{n}$.

$f(\hat{n},\varepsilon,\delta)$ is a nonzero value that rapidly approaches 1. Since f represents a lower bound on $\mathbf{Adv}^{\text{ror-cdla}}_{\Pi,\mathcal{L}_m,\mathcal{A},\mathcal{S}}(\lambda)$, our adversary $\mathcal{A}$ has a non-zero advantage, completing the proof. □

Observe that as n grows large, the lower-bound converges to $\frac{\kappa s_{max}}{\kappa-1}$, which depends solely on the storage footprint since $\kappa = \frac{\sigma}{n s_{max}}$. This highlights a fundamental trade-off between storage and bandwidth footprints: an optimal $\mathcal{ML}_m$ scheme can leverage more storage to lower the bandwidth footprint.

3.2 Padded PANCAKE: Optimal $\mathcal{ML}_m$ scheme

We now present a $\mathcal{ML}_m$ scheme, Padded PANCAKE (Figure 6), defined as PPC := $(\mathsf{Init}_{\text{PPC}}, \mathsf{Query}_{\text{PPC}})$ and adapted from PANCAKE [6] to the variable-length setting. We begin with a summary of PANCAKE's approach and then describe Padded PANCAKE's key differences.

PANCAKE leverages knowledge of the access distribution π to ensure an adversary observes a uniform access distribution over the encrypted KV pairs. First, it selectively replicates KV pairs with high access probability while ensuring the total number of keys in the encrypted store is exactly $2\cdot n$ (adding fake replicas if necessary). This smoothes out the access distribution over the resulting encrypted KV pairs to some extent; the remaining non-uniformity is removed by injecting fake queries. To this end, it computes a fake access distribution π_f over the replicated encrypted KV pairs required to ensure uniform distribution, assuming the proportion of real queries is δ (set to $1/2$ by default). All encrypted queries are sent in batches of size $B=3$ to ensure that the adversary cannot distinguish fake queries from real ones. PANCAKE thus secures the KV store against access pattern attacks with a $3\times$ bandwidth overhead and $2\times$ storage overhead but assumes that the store contains only uniform-size values.

Padded PANCAKE generalizes PANCAKE to support values of arbitrary sizes. At a high level, it pads every key to the largest value size, s_{max}, and then simply applies PANCAKE to the set of padded KV pairs. More specifically, given a total storage footprint $\sigma = \kappa\cdot n\cdot s_{max}$, its initialization function, $\mathsf{Init}_{\text{PPC}}$, creates $\hat{n} = \lfloor \sigma/s_{max} \rfloor = \lfloor \kappa n \rfloor$ values in the output EKV. Of these values, n contain one of each value $v \in$ KV padded to length s_{max} and encrypted. Of the remaining $\hat{n}-n$ values, some hold padded and encrypted replicas of values in KV that have access probability $> \frac{1}{\hat{n}-n+1}$, while the rest are dummy replicas to ensure that the number of values in EKV is exactly $\hat{n}$. $\mathsf{Query}_{\text{PPC}}$ generates a sequence of $q\cdot B$ real and

fake queries to ensure a uniform access distribution across $\hat{k} \in \mathsf{Keys}(\mathsf{EKV})$.

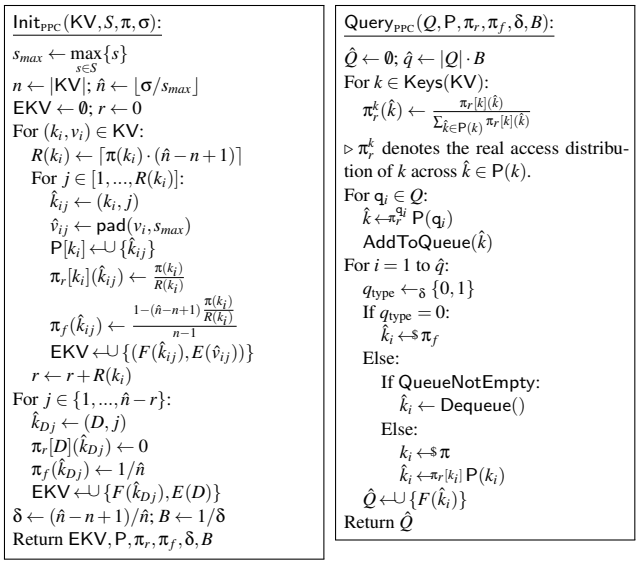


Figure 6: Padded PANCAKE's Init and Query algorithms. The function $\mathsf{pad}(x, s)$ returns string x padded to size s.

Next, we show that PPC is bandwidth-optimal, i.e., it achieves the bandwidth lower bound given by Theorem 3.1.

Theorem 3.2. *For any storage footprint $\sigma = \kappa \cdot n \cdot s_{max}$ ($\kappa \geq 1$),* PPC *is a bandwidth-optimal $\mathcal{ML}_m$ scheme.*

Proof. By construction in Figure 6, $\mathsf{Init}_{\mathrm{PPC}}$ sets $B = \frac{\hat{n}}{\hat{n}-n+1}$, while ensuring π_r, π_f and δ are always valid for any π. As such, PPC has $\beta = B \cdot s_{max} = \frac{\lfloor \kappa n \rfloor \cdot s_{max}}{\lfloor \kappa n \rfloor - n + 1}$, equal to the lower bound given by Theorem 3.1. □

PPC's bandwidth optimality demonstrates the viability of PANCAKE-based solutions in $\mathcal{M}$ — simply padding values in PANCAKE enables an optimal scheme in $\mathcal{ML}_m$ that runs in polynomial time. However, its bandwidth overheads may not be practical for many storage scenarios (*e.g.*, for the Wikipedia dataset with large variance in data sizes, §1.1). We explore relaxed leakage profiles for lower bandwidth footprints next.

4 $\mathcal{L}_S$: Revealing the value sizes

Now we consider the leakage profile $\mathcal{L}_S = (S)$, i.e., the adversary learns the sizes of values in KV. Since schemes in $\mathcal{ML}_S$ do not leak the access distribution (π), their bandwidth and storage footprint must be independent of π. However, leaking S allows us to consider new constructions not in $\mathcal{ML}_m$ — those whose bandwidth and/or storage footprint may depend on S, *e.g.*, the packing example in Figure 2b. We note that it may be possible to leak more information than s_{max} and less than S, *e.g.*, the sum of value sizes ($\sum_i s_i$), as explored in recent work [33]. However, we restrict our focus on leaking S since there are inputs for which $\sum_i s_i$ directly reveals S, *e.g.*, if there are $n-1$ values of size s_{max} and one value of size $s_{max} - 1$, knowing $\sum_i s_i = n \cdot s_{max} - 1$ immediately reveals S. Moreover, our bandwidth lower bounds for leaking S hold for leaking $\sum_i s$. Indeed, our findings complement the recent work, which investigates size-based leakage in greater depth.

4.1 Performance Bounds

The following theorem presents the minimum bandwidth footprint for any scheme $\Pi \in \mathcal{ML}_S$ with storage footprint $\sigma = \kappa \cdot n \cdot s_{max}$. Our proof is similar to that of Theorem 3.1: since a scheme in $\mathcal{ML}_S$ must have the same bandwidth footprint independent of π, we construct a worst-case π to maximize the bandwidth footprint, giving us a lower-bound. However, the relaxed leakage profile allows this lower-bound to be lower than that of Theorem 3.1, depending on how well values in KV can be packed into s_{max}-sized bins.

Theorem 4.1. *Let $N_{OPT} \leq n$ be the minimum number of s_{max}-sized bins needed to bin-pack all values in* KV. *Any $\Pi \in \mathcal{ML}_S$ with storage footprint $\sigma = \kappa \cdot n \cdot s_{max}$ ($\kappa > 0$) and $\hat{s} = s_{max}$ or $\hat{s} \geq \frac{\lfloor \kappa n \rfloor \cdot s_{max}}{\lfloor \kappa n \rfloor - N_{OPT} + 1}$ must have bandwidth footprint $\beta \geq \frac{\lfloor \kappa n \rfloor}{\lfloor \kappa n \rfloor - N_{OPT} + 1} \cdot s_{max}$.*

Proof. Suppose for contradiction that Π has storage footprint $\sigma = \kappa \cdot n \cdot s_{max}$ and bandwidth footprint $\beta = (1 - \delta) \cdot \frac{\lfloor \kappa n \rfloor}{\lfloor \kappa n \rfloor - N_{OPT} + 1} \cdot s_{max}$ for some $0 < \delta < 1$. We will show that the scheme cannot be $\mathcal{L}_S$-secure. Since our proof approach is similar to the proof of Theorem 3.1, we omit common aspects while outlining the salient differences.

Let $\mathcal{L}_S$-adversary $\mathcal{A}$ be defined as in Figure 3 with a single modification: instead of $S \leftarrow \{s_{max}, \ldots, s_{max}\}$, $\mathcal{A}_1$ sets $S \leftarrow \{s_1, \ldots, s_n\}$, which is a list of arbitrary sizes sorted in descending order. The order ensures that the key with the largest value size has access probability $1 - \varepsilon$, which is crucial to $\mathcal{A}$'s approach. We will show that given the large, heavily accessed key and limited range for $\hat{s}$, Π cannot maintain a uniform access distribution over EKV with the assumed bandwidth footprint in Step 1. $\mathcal{A}_2$ can then use the same threshold test from Figure 3 to distinguish between $\text{ROR-CDLA}^0_{\Pi, \mathcal{L}_S, \mathcal{A}, q}(\lambda)$ and $\text{ROR-CDLA}^1_{\Pi, \mathcal{L}_S, \mathcal{A}, \mathcal{S}, q}(\lambda)$ in Step 2.

Step 1: Given $\mathcal{A}$'s choice of inputs, Π's output EKV has a non-uniform access distribution. $\mathcal{A}_1$ generates KV of variable-sized values, where the largest value is also most heavily accessed. The key for this value (say, h) has access probability $1 - \varepsilon$, and the remaining $n-1$ keys have access probability $\frac{\varepsilon}{n-1}$. We will show that one of the keys in EKV that maps to h must have access probability $> 1/\hat{n}$.

Since $\Pi \in \mathcal{M}$, by Property 1, each value $\hat{v} \in \mathsf{Values}(\mathsf{EKV})$ has size $\hat{s}$ and by Property 2, $|\mathsf{EKV}| = \hat{n} \leq \kappa n s_{max} / \hat{s}$. To allow Π to use as much storage as possible, we set $\hat{n} = \lfloor \kappa n s_{max} / \hat{s} \rfloor$. For Property 3, Π maps each $k \in \mathsf{Keys}(\mathsf{KV})$ to a nonempty

set $\mathsf{P}(k) \subseteq \mathsf{Keys}(\mathsf{EKV})$. We focus on the set $\mathsf{P}(h)$ of keys in EKV that h is assigned to; following the same reasoning as Inequality 1 in Theorem 3.1, Property 5 allows us to show:

$$B \geq \frac{1-\varepsilon}{|\mathsf{P}(h)| \cdot p} \tag{4}$$

where p is the equal probability of accessing any key $\hat{h} \in \mathsf{P}(h)$. Again, minimizing bandwidth footprint (which is proportional to B) requires minimizing $|\mathsf{P}(h)|$, and we consider the following two cases for it:

Case 1: $\hat{s} \geq \frac{\lfloor \kappa n \rfloor \cdot s_{max}}{\lfloor \kappa n \rfloor - N_{OPT}+1}$**.** Π's bandwidth footprint is:

$$\beta = B \cdot \hat{s} = \frac{(1-\delta) \cdot \lfloor \kappa n \rfloor \cdot s_{max}}{\lfloor \kappa n \rfloor - N_{OPT} + 1} \tag{5}$$

so we have:

$$B = \frac{(1-\delta)}{\hat{s}} \cdot \frac{\lfloor \kappa n \rfloor \cdot s_{max}}{\lfloor \kappa n \rfloor - N_{OPT} + 1} \leq (1-\delta) < 1$$

violating the constraint $B \geq 1$, making this case impossible.

Case 2: $\hat{s} = s_{max}$**.** Then $\hat{n} = \lfloor \kappa n \rfloor$. Since h has size s_{max}, the values of each $\hat{h} \in \mathsf{P}(h)$ may only contain the value corresponding to h, and nothing else. Then to maximize $|\mathsf{P}(h)|$, Π must minimize the number of values that contain values of the remaining keys $\mathsf{Keys}(\mathsf{KV}) \setminus \{h\}$. This minimum number is equal to the optimal number of s_{max}-sized bins needed to bin-pack the values of $\mathsf{Keys}(\mathsf{KV}) \setminus \{h\}$, which is $N_{OPT} - 1$. Therefore, $|\mathsf{P}(h)| \leq \hat{n} - (N_{OPT} - 1) = \lfloor \kappa n \rfloor - N_{OPT} + 1$. Substituting this into (4), we get:

$$p \geq \frac{1-\varepsilon}{B(\lfloor \kappa n \rfloor - N_{OPT} + 1)}$$

Rearranging (5) gives us $\frac{1}{\lfloor \kappa n \rfloor (1-\delta)} = \frac{1}{B(\lfloor \kappa n \rfloor - N_{OPT}+1)}$ for $\hat{s} = s_{max}$, which simplifies the above lower bound on p to:

$$p \geq \frac{1}{\lfloor \kappa n \rfloor} \cdot \frac{1-\varepsilon}{1-\delta} = \frac{1}{\hat{n}} \cdot \frac{1-\varepsilon}{1-\delta}$$

EKV then contains a key with access probability $p \geq \frac{1}{\hat{n}} \cdot \frac{1-\varepsilon}{1-\delta}$, which is the same result as in Theorem 3.1. While the value of $\hat{n}$ may differ, $\mathcal{A}$ can still choose an ε such that $p > \frac{1}{\hat{n}}$, giving a non-uniform distribution. Once again, a computationally unbounded adversary can distinguish between Π's non-uniform access distribution and a uniform access distribution with a sufficiently-long $\hat{Q}$. However, ROR-CDLA security requires $\mathcal{A}_2$ (Figure 3) to distinguish between the distributions in polynomial time. Specifically, we must show that $\mathcal{A}$ has non-negligible advantage $\mathbf{Adv}^{\text{ror-cdla}}_{\Pi,\mathcal{L}_S,\mathcal{A},\mathcal{S}}(\lambda)$ against Π for all simulator algorithms $\mathcal{S}$. Fortunately, the approach for showing this is identical to Step 2 in the proof of Theorem 3.1, since $\mathcal{A}_2$ uses the exact same threshold-based test; we avoid repeating it for brevity. □

Theorem 4.1 imposes a bandwidth lower bound on $\mathcal{ML}_S$ schemes, but only for $\hat{s} = s_{max}$ and $\hat{s} \geq \frac{\lfloor \kappa n \rfloor \cdot s_{max}}{\lfloor \kappa n \rfloor - N_{OPT}+1}$. The bound does not hold between these two sizes, as values in EKV are large enough to benefit from a more strategic placement of KV values across EKV values. We confirm this intuition with a simple counterexample.

Lemma 3. *There exists a scheme* $\Pi \in \mathcal{ML}_S$ *with* $s_{max} < \hat{s} < \frac{\lfloor \kappa n \rfloor \cdot s_{max}}{\lfloor \kappa n \rfloor - N_{OPT}+1}$ *and input* $(\mathsf{KV}, S, \pi, \sigma)$ *for which* Π *has bandwidth footprint* $\beta < \frac{\lfloor \kappa n \rfloor}{\lfloor \kappa n \rfloor - N_{OPT}+1} \cdot s_{max}$.

Proof. Let $\mathsf{Keys}(\mathsf{KV}) := \{k_1, k_2\}$ with value sizes $S := \{10, 1\}$ and $\sigma = 22$, i.e., $\kappa n = \sigma / s_{max} = 2.2$. The minimum number of s_{max}-sized bins needed to bin-pack the values of KV is $N_{OPT} = 2$, so Theorem 4.1 gives the bandwidth lower bound $\beta = \frac{\lfloor \kappa n \rfloor}{\lfloor \kappa n \rfloor - N_{OPT}+1} \cdot s_{max} = 20$.

Now we give a scheme Π with $10 < \hat{s} < 20$ and bandwidth footprint $\beta < 20$ for all π. Let Π have $\hat{s} = 11$, making $\hat{n} = 2$. Let each value in EKV contain a replica of both values in KV. Then Π can equally divide access probabilities across EKV, i.e., $\forall k \in \mathsf{Keys}(\mathsf{KV}), \forall \hat{k} \in \mathsf{Keys}(\mathsf{EKV}), \pi_r[k](\hat{k}) = \pi(k)/2$, which means both keys in EKV have access probability $1/2$. No additional queries are needed to smooth the access distribution, so Π can set $B = 1$, making $\beta = B \cdot \hat{s} = 11 < 20$. □

In this example, a small increase in $\hat{s}$ above s_{max} vastly improves space utilization. The possibility of this occurrence depends on S, $\hat{s}$, and the corresponding bin-packing. For large n, however, such cases should be rare, and Theorem 4.1's bound should hold for most $\hat{s}$.

Next, we introduce a polynomial-time $\mathcal{ML}_S$ scheme that achieves the bound outlined in Theorem 4.1 for $\hat{s} = s_{max}$.

4.2 Stuffed PANCAKE: Optimal $\mathcal{ML}_S$ scheme

Our proposed scheme in $\mathcal{ML}_S$ exploits revealing S by bin-packing the values in KV into equal-sized bins, and then using PANCAKE to equalize access probabilities across bins.

$\underline{\mathsf{Init}_{\text{SPC}}(\mathsf{KV}, S, \pi, \sigma)}$:
$n \leftarrow |\mathsf{KV}|; s_{max} \leftarrow \max_{s \in S}\{s\}; \mathsf{EKV} \leftarrow \emptyset$
$\hat{s} \leftarrow s_{max}; \hat{n} \leftarrow \lfloor \sigma / \hat{s} \rfloor; \sigma' \leftarrow \hat{n} \cdot \hat{s}$
$\mathsf{KV}', \mathsf{P}' \leftarrow \mathsf{OptBinPack}(\mathsf{KV}, \hat{s})$
For $(k', v') \in \mathsf{KV}'$:
  $\pi_p(k') \leftarrow \sum_{k : k' \in \mathsf{P}'(k)} \pi(k)$
  $S' \xleftarrow{\cup} \{|v'|\}$
$\mathsf{EKV}, \mathsf{P}'', \pi'_r, \pi_f, \delta, B \leftarrow \mathsf{Init}_{\text{PPC}}(\mathsf{KV}', S', \pi_p, \sigma')$
For $k \in \mathsf{Keys}(\mathsf{KV})$:
  $k' \leftarrow \mathsf{P}'[k]$
  $\mathsf{P}(k) \xleftarrow{\cup} \mathsf{P}''[k']$
  For $\hat{k} \in \mathsf{P}''[k']$:
    $\pi_r[k](\hat{k}) \leftarrow \pi(k) / |\mathsf{P}(k)|$
Return $\mathsf{EKV}, \mathsf{P}, \pi_r, \pi_f, \delta, B$

$\underline{\mathsf{Query}_{\text{SPC}}(\ldots)}$:
Return $\mathsf{Query}_{\text{PPC}}(\ldots)$

Figure 7: Stuffed PANCAKE pseudocode. We omit $\mathsf{Query}_{\text{GPC}}$'s arguments since they are the same as $\mathsf{Query}_{\text{PPC}}$'s.

We define Stuffed PANCAKE (Figure 7) as $\text{SPC} := (\mathsf{Init}_{\text{SPC}}, \mathsf{Query}_{\text{SPC}})$, where $\mathsf{Init}_{\text{SPC}}$ first runs an optimal packing algorithm to place n values into $\hat{n}$ bins of size s_{max}. It

then leverages $\mathsf{Init}_{\text{PPC}}$ with storage footprint $\sigma = \lfloor \kappa n \rfloor \cdot s_{max}$ to generate the final $\mathsf{EKV}, \pi_f, \delta, B$, and post-processes outputs P and π_r to ensure correct mappings from keys in KV to keys in EKV. $\mathsf{Query}_{\text{SPC}}$ simply uses $\mathsf{Query}_{\text{PPC}}$ to generate queries to EKV. Note that while we use $\hat{s} = s_{max}$ in our description of SPC, it can easily use $\hat{s} > s_{max}$. We now show that SPC achieves Theorem 4.1's lower bound.

Theorem 4.2. *For any storage footprint $\sigma = \kappa \cdot n \cdot s_{max}$ ($\kappa \geq 1$), SPC is a bandwidth-optimal $\mathcal{ML}_S$ scheme.*

Proof. In $\mathsf{Init}_{\text{SPC}}$, OptBinPack generates KV' containing N_{OPT} values of size s_{max} each. This KV' is then input to $\mathsf{Init}_{\text{PPC}}$ along with storage $\lfloor \kappa n \rfloor \cdot s_{max}$; since $\mathsf{Query}_{\text{SPC}}$ uses the unmodified $\mathsf{Query}_{\text{PPC}}$, SPC's bandwidth footprint is the same as PPC's, except with $n := N_{OPT}$ and $\sigma := \lfloor \kappa n \rfloor \cdot s_{max}$. Substituting these values into Theorem 3.1, we have that SPC has $\beta = \frac{\hat{n}}{\hat{n} - N_{OPT} + 1} \cdot s_{max} = \frac{\lfloor \kappa n \rfloor}{\lfloor \kappa n \rfloor - N_{OPT} + 1} \cdot s_{max}$, equal to the bound in Theorem 4.1. □

Theorem 4.2 shows that SPC's freedom to leak size distribution permits it to achieve a lower bandwidth footprint compared to PPC for the same storage footprint.

Approximating OptBinPack**.** Optimal bin-packing is an NP-hard problem; however, we can use *near-optimal* approximate algorithms that run in polynomial time, *e.g.*, First-Fit Decreasing (FFD) which uses $\leq \frac{11 N_{OPT} + 6}{9}$ bins [42].

5 $\mathcal{L}_\pi$: Revealing the access distribution

Our next leakage profile is $\mathcal{L}_\pi = (s_{max}, \pi)$. Like $\mathcal{L}_S$, $\mathcal{L}_\pi$ is a relaxation of $\mathcal{L}_m$ but allows new constructions that are distinct from those under both $\mathcal{L}_m$ and $\mathcal{L}_S$. Since schemes in $\mathcal{ML}_\pi$ do not leak value sizes S, their storage and bandwidth footprints must be independent of S; however, they can depend on the access distribution π. For instance, the packing approach of Figure 2b is not in $\mathcal{ML}_\pi$ for the same reason that it is not in $\mathcal{ML}_m$: it reveals information about the value sizes. However, the scheme in Figure 2c is in $\mathcal{ML}_\pi$ because the access distribution can be revealed.

Unlike prior sections, we do not have a closed-form lower-bound on bandwidth footprint for $\mathcal{ML}_\pi$; we instead directly provide a bandwidth-optimal $\mathcal{ML}_\pi$ scheme.

5.1 Greedy PANCAKE: Optimal $\mathcal{ML}_\pi$ scheme

We now present a scheme in $\mathcal{ML}_\pi$ that greedily replicates values with the highest access probabilities, then smooths the access distribution across the replicas with fake queries.

Let Greedy PANCAKE be GPC $:= (\mathsf{Init}_{\text{GPC}}, \mathsf{Batch}_{\text{GPC}})$. Given $\sigma = \kappa \cdot n \cdot s_{max}$ storage, $\mathsf{Init}_{\text{GPC}}$ starts with an array R of n elements, where $R[i] = |\mathsf{P}(k_i)|$ is the total number of keys $\hat{k} \in \mathsf{Keys}(\mathsf{EKV})$ that key $k_i \in \mathsf{KV}$ is mapped to. The access probability of k_i is split equally across the keys in P, i.e., for all $\hat{k} \in \mathsf{P}, \pi_r[k_i](\hat{k}) = \pi(k_i)/R[i]$. Initially $R[i] = 1$ for all k_i; for each of the remaining $\hat{n} - n$ keys in EKV, the scheme assigns to it the key k_i with the highest split access probability $\pi(k_i)/R[i]$ at that point. Like in previous PANCAKE-based schemes, $\mathsf{Init}_{\text{GPC}}$ also computes π_f, δ, B to be used in the query algorithm $\mathsf{Query}_{\text{GPC}} := \mathsf{Query}_{\text{PPC}}$.

$\underline{\mathsf{Init}_{\text{GPC}}(\mathsf{KV}, S, \pi, \sigma)}$:
$n \leftarrow |\mathsf{KV}|$; $s_{max} \leftarrow \max_{s \in S}\{s\}$
$\hat{s} \leftarrow s_{max}$; $\hat{n} \leftarrow \lfloor \sigma / \hat{s} \rfloor$
$\mathsf{EKV} \leftarrow \emptyset$; $r \leftarrow 0$; $R \leftarrow [1, \ldots, 1]$
While $\sum_{i=1}^{n} R[i] < \hat{n}$:
  $h \leftarrow \arg\max_i \pi(k_i)/R[i]$
  $R[h] \leftarrow R[h] + 1$
$\hat{p}_{max} \leftarrow \max_{k_i \in \mathsf{Keys}(\mathsf{KV})} \pi(k_i)/R[i]$
For $i \in [1, \ldots, n]$:
  For $j \in [1, \ldots, R[i]]$:
    $\hat{k}_{ij} \leftarrow (k_i, j)$
    $\hat{v}_i \leftarrow \mathsf{pad}(v_i, s_{max})$
    $\mathsf{P}[k_i] \xleftarrow{\cup} \{\hat{k}_{ij}\}$
    $\pi_r[k_i](\hat{k}_{ij}) \leftarrow \pi(k_i)/R[i]$
    $\pi_f(\hat{k}_{ij}) \leftarrow \frac{\hat{p}_{max} - \pi(k_i)/R[i]}{\hat{n} \cdot \hat{p}_{max} - 1}$
    $\mathsf{EKV} \xleftarrow{\cup} \{(F(\hat{k}_{ij}), E(\hat{v}_i))\}$
$\delta \leftarrow 1/(\hat{n} \cdot \hat{p}_{max})$; $B \leftarrow 1/\delta$
Return $\mathsf{EKV}, \mathsf{P}, \pi_r, \pi_f, \delta, B$

$\underline{\mathsf{Query}_{\text{GPC}}(\ldots)}$:
Return $\mathsf{Query}_{\text{PPC}}(\ldots)$

Figure 8: Greedy PANCAKE's Init & Query algorithms. We omit $\mathsf{Query}_{\text{GPC}}$ arguments since they are the same as $\mathsf{Query}_{\text{PPC}}$'s.

Theorem 5.1 shows that Greedy PANCAKE is bandwidth-optimal if the entire storage footprint $\sigma = \kappa \cdot n \cdot s_{max}$ must be used. We defer the proof to Appendix B but summarize our approach here in two steps. First, since the bandwidth footprint of all schemes in $\mathcal{ML}_\pi$ must be independent of S, by Lemma 2, increasing the size $\hat{s}$ of values in EKV beyond s_{max} is not beneficial to reducing bandwidth footprint for schemes in $\mathcal{ML}_\pi$. Second, we show that Greedy PANCAKE using $\hat{s} = s_{max}$ is bandwidth-optimal using the "greedy stays ahead" argument: for each of the $\hat{n} - n$ keys in EKV to which GPC assigns a key from KV, the resulting maximal split access probability $\hat{p}_{max} = \max_{k_i \in \mathsf{Keys}(\mathsf{KV})} \pi(k_i)/R[i]$ is at most that of an optimal algorithm at the same step.

Theorem 5.1. *For any storage footprint $\sigma = \kappa \cdot n \cdot s_{max}$ ($\kappa \geq 1$), GPC achieves the lowest bandwidth footprint β for any $\mathcal{ML}_\pi$ scheme that uses exactly σ storage.*

Intuitively, since GPC can reveal the access distribution, it allocates storage to values proportional to their access probability, maximally utilizing the given storage footprint to minimize bandwidth footprint. A caveat for Greedy PANCAKE, however, is that it is only bandwidth-optimal among schemes that must use all of the storage given, i.e. *exactly* $\kappa \cdot n \cdot s_{max}$ bytes. There are cases when using storage less than $\kappa \cdot n \cdot s_{max}$ results in a lower bandwidth footprint. Consider $\pi = (\frac{1}{5}, \frac{2}{5}, \frac{2}{5})$ and $\sigma = 6 s_{max}$; GPC creates six replicas, i.e., $\beta \geq \sigma \cdot \hat{p}_{max} = \frac{6}{5} s_{max}$. However, one can get a uniform access distribution over EKV with just five replicas: assign 2 replicas each to the keys with

access probability $\frac{2}{5}$ and 1 to the key with probability $\frac{1}{5}$, giving us $\beta = s_{max}$. With this intuition, we can modify GPC to achieve the minimum β for any $\sigma \leq \kappa \cdot n \cdot s_{max}$:

GPC*. We define $\text{GPC}^* = (\text{Init}_{\text{GPC}^*}, \text{Query}_{\text{GPC}})$, where $\text{Init}_{\text{GPC}^*}$ iterates through storage footprints σ from $n \cdot s_{max}$ to $\kappa \cdot n \cdot s_{max}$. For each σ, GPC* runs $(\mathsf{EKV}, \mathsf{P}, \pi_r, \pi_f, \delta, B) \leftarrow \text{Init}_{\text{GPC}}(\mathsf{KV}, S, \pi, \sigma)$ and tracks a running minimum on bandwidth footprint across the outputs from Init_{GPC}, and returns the $(\mathsf{EKV}, \mathsf{P}, \pi_r, \pi_f, \delta, B)$ for the global minimum.

Theorem 5.2. *For any storage footprint $\sigma = \kappa \cdot n \cdot s_{max}$ ($\kappa \geq 1$), GPC* is a bandwidth-optimal $\mathcal{ML}_{\pi}$ scheme.*

Proof. This follows from Theorem 5.1 and GPC*'s use of GPC in its exhaustive search for the optimal bandwidth footprint over storage footprints from $n \cdot s_{max}$ to $\kappa \cdot n \cdot s_{max}$. □

6 $\mathcal{L}_{S\pi}$: Revealing value sizes & distribution

Our last considered leakage profile is $\mathcal{L}_{S\pi} = (S, \pi)$, where the adversary may know both the value sizes S and access distribution π over KV. As the most relaxed leakage profile considered, $\mathcal{L}_{S\pi}$ allows for a much larger solution space than the previous leakage profiles. For instance, both packing (Figure 2b) and replication (Figure 2c) can be used by schemes in $\mathcal{ML}_{S\pi}$. Unfortunately, the search space for a performance-optimal scheme in $\mathcal{ML}_{S\pi}$ is too large, and we currently do not have a polynomial time algorithm that achieves the performance lower-bound. However, we provide a MILP formulation that yields a bandwidth-optimal construction (§6.1), along with Greedy Stuffed PANCAKE (§6.2), a sub-optimal poly-time algorithm in $\mathcal{ML}_{S\pi}$ that is efficient in practice.

6.1 MILP-based optimal $\mathcal{ML}_{S\pi}$ scheme

We formulate the problem of assigning values $v \in \mathsf{Values}(\mathsf{KV})$ and access probabilities to values $\hat{v} \in \mathsf{Values}(\mathsf{EKV})$ to minimize bandwidth footprint as an MILP.

For a key $k \in \mathsf{Keys}(\mathsf{KV})$, let $\hat{p}_{k,\hat{k}}$ be the portion of its real access probability $\pi(k)$ assigned to $\hat{k} \in \mathsf{Keys}(\mathsf{EKV})$; note that if $\hat{k} \notin \mathsf{P}(k)$ then $\hat{p}_{k,\hat{k}} = 0$, and $\sum_{\hat{k} \in \mathsf{EKV}} \hat{p}_{k,\hat{k}} = \pi(k)$. Let $\hat{p}_{\hat{k}} = \sum_{k \in \mathsf{KV}} \hat{p}_{k,\hat{k}}$, i.e., the summed access probabilities of the replicas of keys $k \in \mathsf{Keys}(\mathsf{KV})$ assigned to a key $\hat{k} \in \mathsf{Keys}(\mathsf{EKV})$. Let $\hat{p}_{max} = \max_{\hat{k} \in \mathsf{EKV}} \hat{p}_{\hat{k}}$, and $\hat{k}^*$ be the corresponding key in $\mathsf{Keys}(\mathsf{EKV})$. The objective function for our MILP is given below. We prove that it lower-bounds the bandwidth footprint in Lemma 5 (Appendix B).

$$\textbf{minimize } \hat{n} \cdot \hat{s} \cdot \hat{p}_{max} \tag{6}$$

Our goal is subject to the following constraints:

(a) The real access probabilities must be between 0 and 1.

(b) The real access probabilities across the replicas of a key $k \in \mathsf{KV}$ must sum to k's real access probability.

(c) A key $k \in \mathsf{KV}$ assigned to $\hat{k} \in \mathsf{EKV}$ via P should have non-zero access probability only if $\hat{k} \in \mathsf{P}(k)$.

(d) Sum of value sizes $s_i \in S$ corresponding to keys $k_i \in \mathsf{KV}$ assigned to $\hat{k} \in \mathsf{EKV}$ via P must not exceed $\hat{s}$.

Let $y_{k,\hat{k}} \in \{0,1\}$ be a decision variable that denotes whether or not key $k \in \mathsf{KV}$ mapped to key $\hat{k} \in \mathsf{EKV}$ via P. Then the above constraints can be encoded in the MILP as follows:

$$\forall k \in \mathsf{KV}, \hat{k} \in \mathsf{EKV}, \quad 0 \leq \hat{p}_{k,\hat{k}} \leq 1 \tag{a}$$

$$\forall k \in \mathsf{KV}, \quad \sum_{\hat{k} \in \mathsf{EKV}} \hat{p}_{k,\hat{k}} = \pi(k) \tag{b}$$

$$\forall k \in \mathsf{KV}, \hat{k} \in \mathsf{EKV}, \quad \hat{p}_{k,\hat{k}} \leq y_{k,\hat{k}} \cdot \pi(k) \tag{c}$$

$$\forall \hat{k} \in \mathsf{EKV}, \quad \sum_{k_i \in \mathsf{KV}} y_{k_i,\hat{k}} \cdot s_i \leq \hat{s} \tag{d}$$

The above formulation is not linear: since $\hat{p}_{max}$ is the maximum of $\hat{n}$ linear functions of decision variables, the goal (6) contains a min-max term. However, this term can be linearized through standard techniques [43], making it solvable. Also, since the formulation does not ensure Property 4, we adapt our MILP scheme below to fulfill this requirement.

For each storage capacity $\kappa \cdot n \cdot s_{max}$ and feasible pack size $\hat{s}$ ranging from s_{max} to $s_{tot} = \sum_i s_i$ the $\text{Init}_{\text{MILP}}$ runs a program solver $\mathsf{Solve}(\cdot)$ on the problem formulation. After choosing the bandwidth-optimal pack size, $\text{Init}_{\text{MILP}}$ computes $(\mathsf{EKV}, \mathsf{P}, \pi_r)$ according to the solver output, and (π_f, δ, B) based on Init_{PPC}. Like GPC (§5.1), MILP uses exactly the storage footprint σ given. As with GPC, this has a simple fix: run $\text{Init}_{\text{MILP}}$ for each feasible storage footprint from s_{tot} to σ for a bandwidth-optimal scheme for all $\sigma \leq \kappa \cdot n \cdot s_{max}$. The algorithm definitions are deferred to the full version [44].

Theorem 6.1. *For any storage footprint $\sigma = \kappa \cdot n \cdot s_{max}$ ($\kappa > 0$), MILP is a bandwidth-optimal $\mathcal{ML}_{S\pi}$ scheme.*

Proof. Optimality follows from the objective function of MILP, while feasibility follows from its constraints. □

While the MILP enables an optimal $\mathcal{ML}_{S\pi}$ solution, it is an NP-hard problem. We show a sub-optimal but polynomial-time construction in $\mathcal{ML}_{S\pi}$ next.

6.2 PANCAKE-based poly-time $\mathcal{ML}_{S\pi}$ scheme

Greedy Stuffed PANCAKE (GSPC) is a combination of SPC and GPC*: $\text{Init}_{\text{GSPC}}$ first packs the values of KV into s_{max}-sized bins, then passes the result to $\text{Init}_{\text{GPC}^*}$, while $\text{Query}_{\text{GSPC}}$ is the same as $\text{Query}_{\text{GPC}^*}$ (see the full version [44] for pseudocode). While GSPC runs in polynomial time, it is sub-optimal:

Lemma 4. *There exists a scheme $\Pi \in \mathcal{ML}_{S\pi}$ and input $(\mathsf{KV}, S, \pi, \sigma)$ for which Π has bandwidth footprint β lower than GSPC's.*

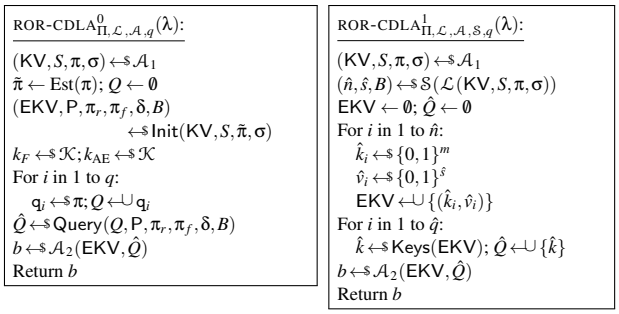


Figure 9: ROR-CDLA security game for key-value store schemes.

Proof. Let the input have sizes $S = \{4,3,2,1\}$, access distribution $(0.1,0.1,0.3,0.5)$, and storage footprint $\sigma = 16$.

First, GSPC sets $\hat{s} = 4$ and bin-packs the values into $N_{OPT} = 3$ bins $(\{4\},\{3\},\{2,1\})$ with access probabilities $(0.1,0.1,0.8)$. GSPC then runs $\mathsf{Init}_{\mathrm{GPC}^*}$, which replicates the last bin, giving $(\{4\},\{3\},\{2,1\},\{2,1\})$ with access probabilities $(0.1,0.1,0.4,0.4)$. Then GSPC sets $B = \hat{n} \times 0.4 = 1.6$ to get $\beta = B \cdot \hat{s} = 6.4$.

Now consider an alternate scheme Π that makes the following assignment, where each tuple (s,p) denotes a value of size s with access probability p, and separate values in EKV are delineated by $\{\}$: $\{(4,0.1)\}$, $\{(1,0.2),(3,0.1)\}$, $\{(1,0.15),(2,0.15)\}$,$\{(1,0.15),(2,0.15)\}$. Since $\hat{p}_{max} = 0.3$, Π can set $B = \hat{n} \times 0.3 = 1.2$ to get $\beta = 4.8 < 6.4$. □

The counterexample highlights the key shortcomings of GSPC: it cannot (i) replicate *individual* values in KV, or (ii) perform *non-uniform* load balancing across replicas. Nevertheless, as shown in §1.1 GSPC is bandwidth-efficient for large n.

7 Formal Security Analysis using ROR-CDLA

We present our new security model ROR-CDLA, which builds on prior work [6] to capture schemes in $\mathcal{M}$ as defined in §2.2. It extends prior work's ROR-CDA model [6], which captures security against access pattern attacks by a passive persistent adversary without accounting for length leakage attacks.

Technical preliminaries. All algorithms including adversaries are assumed to run in time polynomial in the security parameter, denoted λ. A scheme is secure if the advantage gained by any adversary is negligible, i.e., tends to zero faster than the inverse of any polynomial.

Definition 1 (ROR-CDLA with simulator)**.**

Let $\Pi = (\mathsf{Init},\mathsf{Query})$ be an encrypted storage scheme as defined in §2 and let $\mathcal{L}$ be a leakage function. For algorithms $\mathcal{A}$ and $\mathcal{S}$, we define the two games:

ROR-CDLA$^0_{\Pi,\mathcal{L},\mathcal{A},q}(\lambda)$ $\mathcal{A}(1^\lambda)$ picks inputs $(\mathsf{KV},S,\pi,\sigma)$. The game then runs $(\mathsf{EKV},\mathsf{P},\pi_r,\pi_f,\delta,B) \leftarrow\$ \mathsf{Init}(\mathsf{KV},S,\pi,\sigma)$, samples q queries (Q) from π, and runs $\hat{Q} \leftarrow \mathsf{Query}(Q,\mathsf{P},\pi_r,\pi_f,\delta,B)$. It gives $(\mathsf{EKV},\hat{Q})$ to $\mathcal{A}$, which eventually returns a bit that the game outputs.

ROR-CDLA$^1_{\Pi,\mathcal{L},\mathcal{A},\mathcal{S},q}(\lambda)$ $\mathcal{A}(1^\lambda)$ picks inputs $(\mathsf{KV},S,\pi,\sigma)$. The game then runs $(\hat{n},\hat{s},B) \leftarrow\$ \mathcal{S}(\mathcal{L}(\mathsf{KV},S,\pi,\sigma))$ and generates an EKV consisting of $\hat{n}$ random bit strings of size $\hat{s}$. It samples $\hat{Q}$ as $\hat{q} = qB$ queries from a uniform distribution over the keys of EKV, and gives $(\mathsf{EKV},\hat{Q})$ to $\mathcal{A}$, which eventually returns a bit that the game outputs.

We define the ROR-CDLA-$\mathcal{L}$-advantage of $\mathcal{A}$ and $\mathcal{S}$ to be

$$\mathbf{Adv}^{\text{ror-cdla}}_{\Pi,\mathcal{L},\mathcal{A},\mathcal{S}}(\lambda) = |\Pr[\text{ROR-CDLA}^0_{\Pi,\mathcal{L},\mathcal{A},q}(\lambda) = 1] - \Pr[\text{ROR-CDLA}^1_{\Pi,\mathcal{L},\mathcal{A},\mathcal{S},q}(\lambda) = 1]|$$

and we say that Π is ROR-CDLA-$\mathcal{L}$-secure if for all polynomial-time adversaries $\mathcal{A}$ there exists an algorithm $\mathcal{S}$ such that $\mathbf{Adv}^{\text{ror-cdla}}_{\Pi,\mathcal{L},\mathcal{A},\mathcal{S}}(\lambda)$ is negligible.

Note that in the real world, Init for schemes in $\mathcal{M}$ must use an estimate $\tilde{\pi}$ of π, though we define our constructions on input π for simplicity. We will prove that our schemes are secure if the estimate is sufficiently accurate, i.e., $\tilde{\pi}$ is indistinguishable from π with a limited number of samples.

Leakage	ROR-CDA $\Rightarrow \mathcal{ML}_{(\cdot)}$?	$\mathcal{ML}_{(\cdot)} \Rightarrow$ ROR-CDA?
$\mathcal{L}_m$	✗	✓
$\mathcal{L}_S$	✗	✓
$\mathcal{L}_\pi$	✗	✗
$\mathcal{L}_{S\pi}$	✗	✗

Figure 10: **ROR-CDA vs ROR-CDLA.** The second column shows for each leakage profile $\mathcal{ML}_{(\cdot)}$ whether or not any ROR-CDA-secure scheme is in $\mathcal{ML}_{(\cdot)}$. The third column shows whether any scheme in $\mathcal{ML}_{(\cdot)}$ is also ROR-CDA-secure.

Comparison to ROR-CDA. While ROR-CDLA builds on ROR-CDA, it captures a distinct security goal. As shown in Figure 10, no ROR-CDA scheme is ROR-CDLA-secure under any leakage profile: unlike ROR-CDA, ROR-CDLA security requires hiding individual value sizes during access.

For $\mathcal{L}_m$ and $\mathcal{L}_S$, all ROR-CDLA schemes are ROR-CDA-secure as they hide the access distribution π, revealing only value size related information (s_{max} and S respectively) during initialization. The similarity of the definitions means that $\mathcal{L}_m$- and $\mathcal{L}_S$-secure schemes can be derived from an ROR-CDA scheme such as PANCAKE with minor transformations (i.e., to hide value sizes). The opposite holds for π-leaking profiles $\mathcal{L}_\pi$ and $\mathcal{L}_{S\pi}$, as ROR-CDA requires hiding access distribution during both access and initialization. While our secure constructions borrow selective replication and fake querying techniques from PANCAKE, they do not directly use PANCAKE. We prove the security of our constructions across §3.2–§6.2 in the full version of our paper [44].

8 Discussion & Future Work

Our work opens up several directions for future research:

Dynamic distributions. Our analysis focused on data stores with static access patterns and value size distributions. While our lower bounds still hold for dynamic distributions, finding upper bounds is an important next step. For example, an adaptive alternative to MILP would be valuable, as MILP currently requires complete re-initialization should the KV store's access or size distribution change. PANCAKE's [6] adaptations to dynamic access distributions provide a starting point, but additional study is required for dynamic value sizes.

Performance bounds for efficient schemes. In §6.2 we introduced Greedy Stuffed PANCAKE scheme as a polynomial-time approximation of MILP that always finds a feasible, but possibly suboptimal, solution. An analysis of whether its performance is within a bounded factor of the optimal would be interesting for future research.

Completeness of bounds. As Lemma 3 has shown, the lower bound on bandwidth footprint for $\mathcal{ML}_S$ does not hold for $s_{max} < \hat{s} < \frac{\lfloor \kappa n \rfloor}{\lfloor \kappa n \rfloor - N_{OPT} + 1} \cdot s_{max}$. This leaves a gap for which the lower bound on bandwidth footprint, as well as an optimal $\mathcal{ML}_S$ scheme, is unknown. We believe multi-dimensional bin-packing approaches, which take both access probability and value sizes into account, could provide answers for both.

Active adversaries. Our constructions allow us to characterize the storage, bandwidth, and security tradeoff in $\mathcal{M}$. However, $\mathcal{M}$ excludes active adversaries from our threat model. While our lower bounds for passive persistent adversaries also hold for active adversaries, our proposed constructions would require additional protections against injected queries. Applying length leakage analysis to oblivious access schemes designed for this stronger threat model, such as ORAMs, would be a promising direction.

Compression. $\mathcal{M}$ rules out schemes that reduce the value sizes in EKV via compression. However, such schemes may improve throughput at lower storage footprints and even achieve lower bandwidth than our bounds. Extending our analysis to compression schemes will further expand the discussion on the storage-bandwidth-security tradeoff.

9 Conclusion

We explored the problem of hiding length leakage in oblivious data access mechanisms with passive persistent adversaries. We developed novel security models with different leakage profiles and strong lower bounds on achievable storage and bandwidth footprint under them. We also presented provably secure schemes for each profile that are bandwidth-optimal.

Acknowledgments

We thank the anonymous reviewers and the shepherd for their valuable feedback on our paper. We also thank Thomas Ristenpart for many insightful discussions during this work. Our research is supported in part by NSF Awards CNS-2054957, CNS-2047220, CNS-2147946 and CNS-2112562.

References

[1] Mohammad Saiful Islam, Mehmet Kuzu, and Murat Kantarcioglu. Access pattern disclosure on searchable encryption: Ramification, attack and mitigation. In *NDSS*, 2012.

[2] Georgios Kellaris, George Kollios, Kobbi Nissim, and Adam O'Neill. Generic attacks on secure outsourced databases. In *ACM CCS*, 2016.

[3] David Cash, Paul Grubbs, Jason Perry, and Thomas Ristenpart. Leakage-abuse attacks against searchable encryption. In *ACM CCS*, 2015.

[4] Evgenios Kornaropoulos, Charalampos Papamanthou, and Roberto Tamassia. Data recovery on encrypted databases with k-nearest neighbor query leakage. In *IEEE S&P*, 2019.

[5] Oded Goldreich and Rafail Ostrovsky. Software protection and simulation on oblivious rams. *JACM*, 1996.

[6] Paul Grubbs, Anurag Khandelwal, Marie-Sarah Lacharité, Lloyd Brown, Lucy Li, Rachit Agarwal, and Thomas Ristenpart. Pancake: Frequency smoothing for encrypted data stores. In *USENIX Security*, 2020.

[7] Natacha Crooks, Matthew Burke, Ethan Cecchetti, Sitar Harel, Rachit Agarwal, and Lorenzo Alvisi. Obladi: Oblivious serializable transactions in the cloud. In *USENIX OSDI*, 2018.

[8] Cetin Sahin, Victor Zakhary, Amr El Abbadi, Huijia Lin, and Stefano Tessaro. Taostore: Overcoming asynchronicity in oblivious data storage. In *IEEE S&P*, 2016.

[9] Vincent Bindschaedler, Muhammad Naveed, Xiaorui Pan, XiaoFeng Wang, and Yan Huang. Practicing oblivious access on cloud storage: The gap, the fallacy, and the new way forward. In *ACM CCS*, 2015.

[10] Midhul Vuppalapati, Kushal Babel, Anurag Khandelwal, and Rachit Agarwal. SHORTSTACK: Distributed, fault-tolerant, oblivious data access. In *USENIX OSDI*, 2022.

[11] Emma Dauterman, Vivian Fang, Ioannis Demertzis, Natacha Crooks, and Raluca Ada Popa. Snoopy: Surpassing the scalability bottleneck of oblivious storage. In *ACM SOSP*, 2021.

[12] Emil Stefanov and Elaine Shi. Oblivistore: High performance oblivious cloud storage. In *IEEE S&P*, 2013.

[13] Anrin Chakraborti and Radu Sion. Concuroram: High-throughput stateless parallel multi-client ORAM. In *NDSS*, 2019.

[14] Elette Boyle, Kai-Min Chung, and Rafael Pass. Oblivious parallel ram and applications. In *IACR TCC*, 2016.

[15] T-H Hubert Chan, Kartik Nayak, and Elaine Shi. Perfectly secure oblivious parallel ram. In *IACR TCC*, 2018.

[16] T-H Hubert Chan and Elaine Shi. Circuit opram: Unifying statistically and computationally secure orams and oprams. In *IACR TCC*, 2017.

[17] Gareth T Davies, Christian Janson, and Daniel P Martin. Client-oblivious opram. In *ICICS*, 2020.

[18] Peter Williams, Radu Sion, and Alin Tomescu. Privatefs: A parallel oblivious file system. In *ACM CCS*, 2012.

[19] Jacob R. Lorch, Bryan Parno, James Mickens, Mariana Raykova, and Joshua Schiffman. Shroud: Ensuring private access to large-scale data in the data center. In *USENIX FAST*, 2013.

[20] Elette Boyle and Moni Naor. Is there an oblivious RAM lower bound? In *ACM ITCS*, 2016.

[21] Kasper Green Larsen and Jesper Buus Nielsen. Yes, there is an oblivious ram lower bound! In *IACR CRYPTO*, 2018.

[22] Giuseppe Persiano and Kevin Yeo. Lower bounds for differentially private rams. In *IACR EUROCRYPT*, 2019.

[23] Mor Weiss and Daniel Wichs. Is there an oblivious RAM lower bound for online reads? In Amos Beimel and Stefan Dziembowski, editors, *IACR TCC*, 2018.

[24] Kasper Green Larsen, Mark Simkin, and Kevin Yeo. Lower bounds for multi-server oblivious rams. In *IACR TCC*, 2020.

[25] Sarvar Patel, Giuseppe Persiano, and Kevin Yeo. What storage access privacy is achievable with small overhead? In *ACM PODS*, 2019.

[26] Berk Atikoglu, Yuehai Xu, Eitan Frachtenberg, Song Jiang, and Mike Paleczny. Workload Analysis of a Large-scale Key-value Store. In *ACM SIGMETRICS Performance Evaluation Review*, 2012.

[27] Andrew C. Reed and Michael K. Reiter. Optimally hiding object sizes with constrained padding, 2021.

[28] Seny Kamara and Tarik Moataz. Computationally volume-hiding structured encryption. In *IACR EUROCRYPT*, 2019.

[29] Min Xu, Armin Namavari, David Cash, and Thomas Ristenpart. Searching encrypted data with size-locked indexes. In *USENIX Security*, 2021.

[30] Twitter 2010 data set. https://www.isi.edu/~lerman/downloads/twitter/twitter2010.html.

[31] Wikimedia Downloads. https://dumps.wikimedia.org/enwiki/20240101/.

[32] Juncheng Yang, Yao Yue, and K. V. Rashmi. A large scale analysis of hundreds of in-memory cache clusters at twitter. In *USENIX OSDI*, 2020.

[33] Evgenios M. Kornaropoulos, Nathaniel Moyer, Charalampos Papamanthou, and Alexandros Psomas. Leakage inversion: Towards quantifying privacy in searchable encryption. In *ACM CCS*, 2022.

[34] Broadcom: Secure Web Gateway. https://www.broadcom.com/products/cybersecurity/network/web-protection.

[35] Simon Oya and Florian Kerschbaum. IHOP: Improved statistical query recovery against searchable symmetric encryption through quadratic optimization. In *USENIX Security*, 2022.

[36] David Cash, Paul Grubbs, Jason Perry, and Thomas Ristenpart. Leakage-abuse attacks against searchable encryption. In *ACM CCS*, 2015.

[37] Paul Grubbs, Marie-Sarah Lacharité, Brice Minaud, and Kenneth G Paterson. Learning to reconstruct: Statistical learning theory and encrypted database attacks. In *IEEE S&P*, 2019.

[38] Crime attack. https://threatpost.com/091312/77006/.

[39] Yoel Gluck, Neal Harris, and Angelo Prado. BREACH: reviving the CRIME attack. *Unpublished*, 2013.

[40] Kevin P. Dyer, Scott E. Coull, Thomas Ristenpart, and Thomas Shrimpton. Peek-a-Boo, I Still See You: Why Efficient Traffic Analysis Countermeasures Fail. In *IEEE S&P*, 2012.

[41] Steven Lambregts, Huanhuan Chen, Jianting Ning, and Kaitai Liang. VAL: Volume and access pattern leakage-abuse attack with leaked documents. In *ESORICS*, 2022.

[42] György Dósa. The Tight Bound of First Fit Decreasing Bin-Packing Algorithm Is $\mathrm{FFD}(I) \leq 11/9\mathrm{OPT}(I) + 6/9$. In *Combinatorics, Algorithms, Probabilistic and Experimental Methodologies*. Springer, 2007.

[43] Boris N. Pshenichnyj. *The linearization method for constrained optimization*, volume 22. Springer, 2012.

[44] Grace Jia, Rachit Agarwal, and Anurag Khandelwal. Length leakage in oblivious data access mechanisms. IACR ePrint, 2024. https://ia.cr/2024/863.

[45] Stéphane Boucheron, Gábor Lugosi, and Pascal Massart. Concentration inequalities - a nonasymptotic theory of independence. In *Concentration Inequalities*, 2013.

[46] Michael Mitzenmacher and Eli Upfal. *Probability and Computing: Randomized Algorithms and Probabilistic Analysis*. Cambridge University Press, 2005.

A Additional Experimental Results

In addition to the Twitter and Wikipedia datasets, we also apply our key results from §1.1 to two workloads from Facebook's Memcached deployments [26]: (i) the USR workload for user account status information, and (ii) the ETC workload for large-scale, general-purpose KV storage.

Figure 11 remains consistent with our theoretical results: all schemes observe a drop in bandwidth overhead as more storage is provided, and the strongest leakage profile incurs the highest bandwidth overhead.

Facebook-USR has a bimodal distribution of small value sizes that differ by only 5 bytes. This allows all leakage profiles to observe $< 10\times$ bandwidth overheads at a mere $1.3\times$ storage overhead. In fact, the size distribution of this dataset is so extreme that revealing it does not bring performance benefits for data with near-uniform sizes; there is no arrangement of values that better utilizes storage compared to simply padding each value to s_{max}. $\mathcal{L}_m$ and $\mathcal{L}_S$ thus have near-identical bandwidth overheads, as do $\mathcal{L}_\pi$ and $\mathcal{L}_{S\pi}$.

Meanwhile, Facebook-ETC has a long-tailed size distribution like Wikipedia's, and exhibits similar bandwidth overhead trends — the minimum storage overhead for $\mathcal{L}_m$ and $\mathcal{L}_\pi$ leakage settings is over $1200\times$, and the bandwidth overhead for $\mathcal{L}_m$ is more than 10^9. Bandwidth overheads under $\mathcal{L}_S$ and $\mathcal{L}_{S\pi}$ are much lower in comparison but still over $1000\times$, since hiding each value's identity requires fetching for each query s_{max} bytes, which in Facebook-ETC is $> 1000\times$ larger than the average value size. This again confirms our intuition that long-tailed size distributions are more suited to $\mathcal{L}_S$ or $\mathcal{L}_{S\pi}$ leakage settings that avoid the costs of hiding value lengths.

As for access distribution skewness, the results match Figure 1: schemes that do not leak the access distribution (PPC and SPC) observe the same bandwidth overhead regardless of access skew across key-value pairs. For schemes that do leak access distribution (GPC and GSPC), bandwidth overheads increase with the access skew due to noise injection to hide the identity of the most popular key.

B Additional Lemmas & Definitions

We present a bound on B for any scheme in $\Pi \in \mathcal{M}$:

Lemma 5. *For any scheme $\Pi \in \mathcal{M}$, let the sum of the real access probabilities of keys $k \in \mathsf{Keys(KV)}$ contained in key $\hat{k} \in \mathsf{Keys(EKV)}$ be $\hat{p}_{\hat{k}} = \sum_{k \in \mathsf{KV}} \pi_r[k](\hat{k})$. Let $\hat{p}_{max} = \max_{\hat{k} \in \mathsf{EKV}}(\hat{p}_{\hat{k}})$; then $B \geq \hat{n} \cdot \hat{p}_{max}$.*

Proof. Since δ is the proportion of real accesses in $\hat{Q}$ and B is the ratio of the number of queries in $\hat{Q}$ compared to Q, we use δ to give a lower bound on B. For Property 4, Π must choose δ and π_f such that the probability of accessing any $\hat{k} \in \mathsf{EKV}$ is $1/\hat{n}$, i.e.,

$$\delta \cdot \hat{p}_{\hat{k}} + (1-\delta) \cdot \pi_f(\hat{k}) = \frac{1}{\hat{n}}$$

$$\text{or, } \delta \cdot \sum_{k \in \mathsf{KV}} \pi_r[k](\hat{k}) + (1-\delta) \cdot \pi_f(\hat{k}) = \frac{1}{\hat{n}}$$

For the probability $\pi_f(\hat{k})$ to have a valid non-negative value in the above equation for any $\hat{k}$, Π must have $\delta \leq \frac{1}{\hat{n} \cdot \sum_{k \in \mathsf{KV}} \pi_r[k](\hat{k})} \leq \frac{1}{\hat{n} \cdot \hat{p}_{max}}$.

Summing Property 5's inequality over all $k \in \mathsf{Keys(KV)}$:

$$E(\text{real queries in } Q) \leq E(\text{real queries in } \hat{Q})$$

Since all queries in Q are real, and the expected number of real accesses in $\hat{Q}$ is $\delta \cdot |\hat{Q}| = \delta \cdot q \cdot B$, we have $q \leq \delta \cdot q \cdot B$, i.e., $B \geq \frac{1}{\delta} \geq \hat{n} \cdot \hat{p}_{max}$. □

We provide the proofs of Lemmas 1 and 2 (§3.1).

Lemma 1. *Any $\Pi \in \mathcal{M}$ with leakage $\mathcal{L}$ and $S \notin \mathcal{L}$ requires storage footprint $\sigma \geq n \cdot s_{\max}$.*

Proof. Let us consider a scheme Π that has storage footprint $\sigma < n \cdot s_{max}$ when possible, i.e., the sum of its input sizes $\sum_i s_i < n \cdot s_{max}$. Against this scheme, we give an $O(n)$-time adversary $\mathcal{A}$ that, for all algorithms $\mathcal{S}$, has advantage $\mathbf{Adv}^{\text{ror-cdla}}_{\Pi,\mathcal{L},\mathcal{A},\mathcal{S}}(\lambda) = 1/2$.

Now we construct $\mathcal{A}$. $\mathcal{A}_1$ chooses value sizes from one of two distributions: $S_0 = \{s_{max}, ..., s_{max}\}$, i.e., all values have size s_{max}; or $S_1 = \{s_{max}, 1, ..., 1\}$, i.e., only one value has size s_{max}, while the rest are size 1. The two size distributions are chosen such that for S_0 and S_1, Π will output EKV_0 and EKV_1 respectively — while EKV_0 must have storage footprint $\sigma \geq n \cdot s_{max}$ to fulfill Property 3, EKV_1 has a lower storage footprint.

$\mathcal{A}_1$ flips a coin $b \sim \{0,1\}$ and assigns value sizes corresponding to S_b. It then outputs a KV with the chosen size distribution and uniform π. $\mathcal{A}_2$ takes as input EKV and $\hat{Q}$. $\mathcal{A}_2$ ignores $\hat{Q}$, but checks if the storage footprint of EKV is less than $n \cdot s_{max}$ bytes (i.e., $b = 1$) or at least $n \cdot s_{max}$ (i.e., $b = 0$). If the storage footprint matches the coin-flip b, $\mathcal{A}_2$ outputs 1. Otherwise, it outputs 0.

Suppose $\mathcal{A}$ is in $\text{ROR-CDLA}^0_{\Pi,\mathcal{L},\mathcal{A},q}(\lambda)$ interacting with the real Π. If $\mathcal{A}$ chooses S_0, Π always outputs an EKV using $n \cdot s_{max}$ bytes of storage. Similarly if $\mathcal{A}$ chooses S_1, Π always outputs an EKV using less storage, meaning $\Pr[\text{ROR-CDLA}^0_{\Pi,\mathcal{L},\mathcal{A},q}(\lambda) = 1] = 1$.

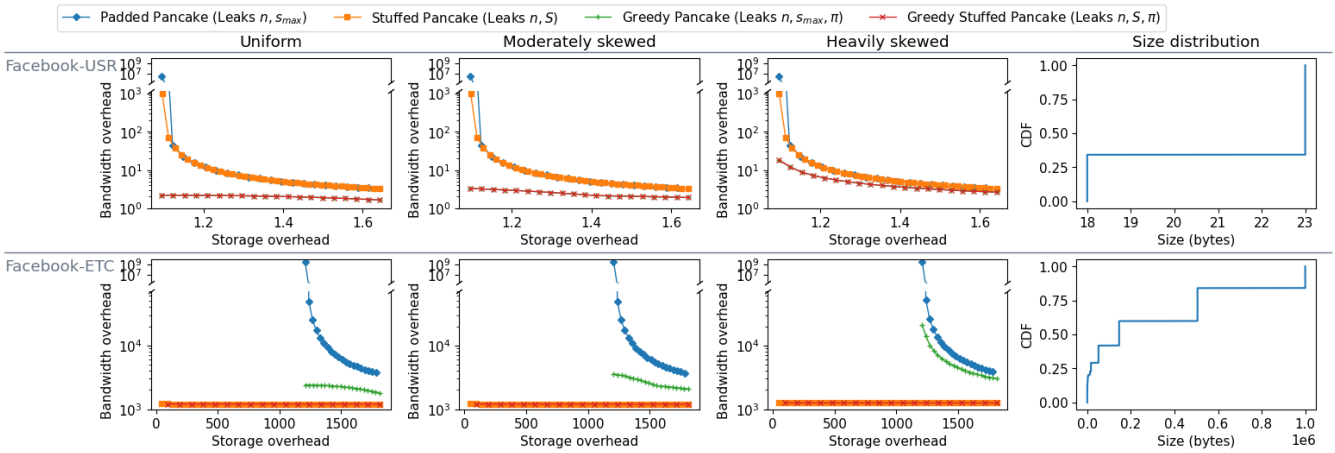


Figure 11: **Bandwidth vs. storage overhead across schemes with different leakage (continued from §1.1).** We use real-world workloads Facebook-USR (top) and Facebook-ETC (bottom) from Facebook Memcached deployments [26], and Zipf access patterns with varying skew (left to right). The rightmost column shows the dataset's object size CDF. Bandwidth overheads for schemes that do not leak object sizes start from the minimum storage footprint required for security, i.e., $n \cdot s_{max}$. The y-axes are in log-scale broken between bandwidth overheads 1200–2e5× (Facebook-USR) and 70000–2e5× (Facebook-ETC).

Now suppose $\mathcal{A}$ is in $\text{ROR-CDLA}^{1}_{\Pi,\mathcal{L},\mathcal{A},\mathcal{S},q}(\lambda)$ interacting with a simulator $\mathcal{S}$. $\mathcal{S}$ only knows the output of $\mathcal{L}_m$ on the KV given by $\mathcal{A}$, which is n and s_{max}. Without access to b, $\mathcal{S}$ can only *guess* whether $\mathcal{A}$ chose S_0 or S_1, and output pack sizes with the storage footprint that $\mathcal{A}$ expects. $\mathcal{S}$ guesses correctly with probability $\frac{1}{2}$, so $\Pr[\text{ROR-CDLA}^{1}_{\Pi,\mathcal{L},\mathcal{A},\mathcal{S},q}(\lambda) = 1] = \frac{1}{2}$.

Thus $\mathcal{A}$'s advantage $\mathbf{Adv}^{\text{ror-cdla}}_{\Pi,\mathcal{L},\mathcal{A},\mathcal{S}}(\lambda) = |1 - \frac{1}{2}| = \frac{1}{2}$. □

Lemma 2. *Given storage footprint $\sigma = \kappa \cdot n \cdot s_{max}$, a scheme $\Pi \in \mathcal{M}$ with leakage $\mathcal{L}$ and $S \notin \mathcal{L}$ that incurs the minimum possible bandwidth footprint must use $\hat{s} = s_{max}$.*

Proof. Let Π be given inputs $\mathsf{KV}, S, \pi, \sigma$. Since Π does not leak S, it must incur the same bandwidth footprint for all S. As such, we consider the worst-case $S = \{s_{max}, \ldots, s_{max}\}$.

By Property 1, each value $\hat{v} \in \mathsf{Values(EKV)}$ has the same size, say $\hat{s} = \gamma \cdot s_{max}$ for some $\gamma \geq 1$. Since all the values $v \in \mathsf{Values(KV)}$ are size s_{max}, we can view each value $\hat{v} \in \mathsf{Values(EKV)}$ as a set of γ "slots", where each slot can hold a copy of an s_{max}-sized input value $v \in \mathsf{Values(KV)}$.

By Property 3, each value in KV maps to at least one s_{max}-sized slot in EKV's values. The goal now is to map each key in KV to one or more slots in a manner that incurs at most β bandwidth footprint. Note that since every $v \in \mathsf{Values(KV)}$ is exactly size s_{max}, non-integer values of γ would only result in wasted space in each value $\hat{v}$ and, therefore, wasted bandwidth per access; as such, we only focus on integral γ values. Since EKV uses total storage $\kappa \cdot n \cdot s_{max}$, the number of values in EKV is $\hat{n} = \frac{\kappa n}{\gamma}$.

Let $\hat{p}_{\hat{k}}$ be the sum of the access probabilities of the keys $k \in \mathsf{Keys(KV)}$ assigned to key $\hat{k} \in \mathsf{Keys(EKV)}$, i.e., $\hat{p}_{\hat{k}} = \sum_{k \in \mathsf{KV}} \pi_r[k](\hat{k})$. Let $\hat{p}_{max} = \max_{\hat{k} \in \mathsf{EKV}} (\hat{p}_{\hat{k}})$. Lemma 5 gives us $B \geq \hat{n} \cdot \hat{p}_{max}$, so we have bandwidth footprint:

$$\begin{aligned}\beta &= B \cdot \gamma \cdot s_{max} \geq \hat{n} \cdot \hat{p}_{max} \cdot \gamma \cdot s_{max} \\ &= \frac{\kappa n}{\gamma} \cdot \hat{p}_{max} \cdot \gamma \cdot s_{max} = \sigma \cdot \hat{p}_{max}\end{aligned}$$

As such, minimizing $\hat{p}_{max}$ minimizes β. Note $\hat{p}_{max}$ is monotonically non-decreasing in γ: larger γ would allow more keys $k \in \mathsf{Keys(KV)}$ to be mapped to keys $\hat{k} \in \mathsf{Keys(EKV)}$ and increase $\hat{p}_{max}$. Thus, the smallest possible value $\gamma = 1$ minimizes bandwidth footprint β. □

B.1 Bandwidth Bound of $\mathcal{ML}_m$

We present the full version of Step 2 of Theorem 3.1's proof, including the derivation of the lower bound on the advantage $\mathbf{Adv}^{\text{ror-cdla}}_{\Pi,\mathcal{L}_m,\mathcal{A},\mathcal{S}}(\lambda)$ of adversary $\mathcal{A}$ as defined in Figure 3.

Step 2: $\mathcal{A}$ has a non-negligible advantage. To show this, we lower-bound $\mathbf{Adv}^{\text{ror-cdla}}_{\Pi,\mathcal{L}_m,\mathcal{A},\mathcal{S}}(\lambda)$ by a function $f(\hat{n}, \varepsilon, \delta)$, and show that with $\varepsilon \to 0$ and sufficiently large values of $\hat{n}$, $f(\hat{n}, \varepsilon, \delta)$ approaches 1 for $0 < \delta < 1$ and $\varepsilon \to 0$. Since

$$\begin{aligned}\mathbf{Adv}^{\text{ror-cdla}}_{\Pi,\mathcal{L}_m,\mathcal{A},\mathcal{S}}(\lambda) = |&\Pr[\text{ROR-CDLA}^{0}_{\Pi,\mathcal{L}_m,\mathcal{A},q}(\lambda) = 1] \\ &- \Pr[\text{ROR-CDLA}^{1}_{\Pi,\mathcal{L}_m,\mathcal{A},\mathcal{S},q}(\lambda) = 1]|,\end{aligned}$$

We obtain this function by finding (i) an upper bound on $\Pr[\text{ROR-CDLA}^{0}_{\Pi,\mathcal{L}_m,\mathcal{A},q}(\lambda) = 1]$, and (ii) a lower bound on $\Pr[\text{ROR-CDLA}^{1}_{\Pi,\mathcal{L}_m,\mathcal{A},\mathcal{S},q}(\lambda) = 1]$. We also note that since $\mathcal{A}_2$'s test does not leverage $\hat{n}$, $\hat{s}$ and B generated by the algorithm $\mathcal{S}$, $\mathcal{A}$ works against all possible $\mathcal{S}$.

Step 2(i): We upper-bound $\Pr[\text{ROR-CDLA}^{0}_{\Pi,\mathcal{L}_m,\mathcal{A},q}(\lambda) = 1]$, i.e., the probability that $\mathcal{A}$ guesses incorrectly that the queries $\hat{Q}$ it is given are sampled from a uniform distribution rather than generated by Π.

To do so, we focus on the key $\hat{h} \in \mathsf{P}(h)$ with access probability p. $\mathcal{A}_2$ outputs 1 if, for all keys in EKV, the number of queries they receive is less than or equal to $\mathcal{A}$'s threshold. We can, therefore, use the probability that $\hat{h}$ receives $\leq (c+3\sqrt{c})\ln\hat{n}$ queries as an upper bound for $\Pr[\text{ROR-CDLA}^0_{\Pi,\mathcal{L}_m,\mathcal{A},q}(\lambda)=1]$, and compute the same using the one-sided Chebyshev inequality, as we show next.

Let $X_{\hat{h}}$ be the random variable that counts the number of queries to $\hat{h}$ in $\hat{Q}$. $X_{\hat{h}}$ follows a binomial distribution $B(\hat{q},p)$ with mean $\mu=\hat{q}p \geq \frac{\hat{q}}{\hat{n}}\cdot\frac{1-\varepsilon}{1-\delta}$ and variance $\sigma^2 = \hat{q}p(1-p) = \hat{q}\left(\frac{1}{\hat{n}}\cdot\frac{1-\varepsilon}{1-\delta}\right)\left(1-\frac{1}{\hat{n}}\cdot\frac{1-\varepsilon}{1-\delta}\right)$. Since $\hat{q}=c\hat{n}\ln\hat{n}$, we have $\mu = c\ln\hat{n}\left(\frac{1-\varepsilon}{1-\delta}\right)$ and $\sigma^2 = \frac{c\ln\hat{n}(1-\varepsilon)(\hat{n}(1-\delta)-(1-\varepsilon))}{\hat{n}(1-\delta)^2}$.

To compute $\Pr[X_{\hat{h}} \leq (c+3\sqrt{c})\ln\hat{n}]$, we apply the one-sided Chebyshev inequality [45]: $\Pr[X_{\hat{h}} \leq \mu - a] \leq \frac{\sigma^2}{\sigma^2+a^2}$. We set:

$$a = \mu - (c+3\sqrt{c})\ln\hat{n} = c\ln\hat{n}\left(\frac{1-\varepsilon}{1-\delta} - 1 - \frac{3}{\sqrt{c}}\right).$$

Then, substituting into Chebyshev's inequality, we have:

$$\begin{aligned}
&\Pr[X_{\hat{h}} \leq (c+3\sqrt{c})\ln\hat{n}] \\
&\leq \frac{\frac{c\ln\hat{n}(1-\varepsilon)(\hat{n}(1-\delta)-(1-\varepsilon))}{\hat{n}(1-\delta)^2}}{\frac{c\ln\hat{n}(1-\varepsilon)(\hat{n}(1-\delta)-(1-\varepsilon))}{\hat{n}(1-\delta)^2} + c^2\ln^2\hat{n}\left(\frac{1-\varepsilon}{1-\delta}-1-\frac{3}{\sqrt{c}}\right)^2} \\
&= \frac{(1-\varepsilon)(\hat{n}(1-\delta)-(1-\varepsilon))}{(1-\varepsilon)(\hat{n}(1-\delta)-(1-\varepsilon)) + \frac{c\ln\hat{n}\left(\frac{1-\varepsilon}{1-\delta}-1-\frac{3}{\sqrt{c}}\right)^2}{\hat{n}(1-\delta)^2}}.
\end{aligned}$$

This gives us our upper-bound:

$$\begin{aligned}
&\Pr[\text{ROR-CDLA}^0_{\Pi,\mathcal{L}_m,\mathcal{A},q}(\lambda)=1] \\
&\leq \frac{(1-\varepsilon)(\hat{n}(1-\delta)-(1-\varepsilon))}{(1-\varepsilon)(\hat{n}(1-\delta)-(1-\varepsilon)) + \frac{c\ln\hat{n}\left(\frac{1-\varepsilon}{1-\delta}-1-\frac{3}{\sqrt{c}}\right)^2}{\hat{n}(1-\delta)^2}}. \qquad (2)
\end{aligned}$$

Note that $\mathcal{A}$ must choose a small enough ε such that $a>0$, specifically $\varepsilon < 1-(1+3\sqrt{c})(1-\delta)$. Since we also need $\varepsilon > 0$ for the π chosen by $\mathcal{A}$ to be a valid distribution, the attack can only work with $\delta > \frac{3}{3+\sqrt{c}}$.

Step 2(ii): We lower-bound $\Pr[\text{ROR-CDLA}^1_{\Pi,\mathcal{L}_m,\mathcal{A},\mathcal{S},q}(\lambda)=1]$, i.e., the probability that $\mathcal{A}$ guesses correctly that the queries in $\hat{Q}$ are sampled from a uniform distribution.

Once again, $\mathcal{A}$ outputs 1 if all keys receive more than $(c+3\sqrt{c})\ln\hat{n}$ queries, so we represent the number of queries to a key in EKV with a binomial random variable. We then use the Chernoff bound to compute the upper bound of the probability that *all* keys receive $> (c+3\sqrt{c})\ln\hat{n}$ queries, which we subtract from 1 to obtain the desired lower bound on $\Pr[\text{ROR-CDLA}^1_{\Pi,\mathcal{L}_m,\mathcal{A},\mathcal{S},q}(\lambda)=1]$, as we show next.

Let $X_{\hat{k}}$ be the random variable that counts the number of queries to $\hat{k} \in \mathsf{Keys}(\mathsf{EKV})$. Each $X_{\hat{k}}$ has a binomial distribution $B(\hat{q},1/\hat{n})$. To find $\Pr[X_{\hat{k}} \leq (c+3\sqrt{c})\ln\hat{n}]$, we apply the Chernoff bound [46]:

$$\Pr[X_{\hat{k}} \geq (1+a)\mu] \leq \exp\left(\frac{-a^2\mu}{3}\right) \text{ for any } 0<a<1$$

Substituting in $a = 3/\sqrt{c}$:

$$\Pr[X_{\hat{k}} \geq (c+3\sqrt{c})\ln\hat{n}] \leq \exp\left(\frac{-\left(\frac{3}{\sqrt{c}}\right)^2 c\ln\hat{n}}{3}\right) = \hat{n}^{-3}$$

By the union bound, the probability that any key receives more than $(c+3\sqrt{c})\ln\hat{n}$ queries is $\leq \hat{n}(\hat{n}^{-3}) = \hat{n}^{-2}$. Thus,

$$\Pr[\text{ROR-CDLA}^1_{\Pi,\mathcal{L}_m,\mathcal{A},\mathcal{S},q}(\lambda)=1] \geq 1-\hat{n}^{-2}. \qquad (3)$$

B.2 Bandwidth-optimality of GPC

Here we prove that Greedy PANCAKE is bandwidth-optimal.

Theorem 5.1. *For any storage footprint $\sigma = \kappa\cdot n\cdot s_{max}$ ($\kappa \geq 1$), GPC achieves the lowest bandwidth footprint β for any $\mathcal{ML}_\pi$ scheme that uses exactly σ storage.*

Proof. Any $\mathcal{ML}_\pi$ scheme must have the same bandwidth footprint for any input sizes S; as such, Lemma 2 holds for schemes in $\mathcal{ML}_\pi$; GPC already uses $\hat{s}=s_{max}$, satisfying this requirement. We show that GPC with $\kappa\cdot n\cdot s_{max}$ storage achieves the optimal bandwidth footprint for $S := \{s_{max},\ldots,s_{max}\}$, and, by extension, all other S.

Let $R[i] = |\mathsf{P}(k_i)|$ be the number of keys $\hat{k} \in \mathsf{Keys}(\mathsf{EKV})$ mapped to key $k_i \in \mathsf{Keys}(\mathsf{KV})$ by GPC, and let $R^*[i]$ be the number of $\hat{k} \in \mathsf{Keys}(\mathsf{EKV})$ mapped to key $k_i \in \mathsf{Keys}(\mathsf{KV})$ by an optimal algorithm. Let $f(j,R) := \max_{k_i\in\mathsf{Keys}(\mathsf{KV})} \pi(k_i)/R[i]$ after j out of $\hat{n}$ keys in EKV have been assigned a key in KV. By induction on j, we show that $f(j,R) \leq f(j,R^*)$ for $n \leq j \leq \hat{n}$.

Base case: $j=n$. Since each key $k \in \mathsf{Keys}(\mathsf{KV})$ is initially assigned exactly one key $\hat{k} \in \mathsf{Keys}(\mathsf{EKV})$, we have:

$$f(j,R) = f(j,R^*) = \max_{k_i\in\mathsf{Keys}(\mathsf{KV})}\{\pi(k_i)\}$$

Induction. Let $f(j,R) \leq f(j,R^*)$ hold for some $j > n$. Consider the $(j+1)^{th}$ key $\hat{k}_{j+1} \in \mathsf{Keys}(\mathsf{EKV})$ to be assigned a key in $\mathsf{Keys}(\mathsf{KV})$. GPC assigns to it $h = \arg\max_{k_i\in\mathsf{Keys}(\mathsf{KV})} \pi(k_i)/R[i]$, which means before this step, $\forall\hat{k} \in \mathsf{P}(h), \pi_r[h](\hat{k}) = f(j,R)$. Since we know $f(j,R) \leq f(j,R^*)$, the best choice that the optimal algorithm has in minimizing $f(j+1,R^*)$ is to assign its $\hat{k}^*_{j+1}$ to the key h^* for which $\forall\hat{k} \in \mathsf{P}(h), \pi_r[h](\hat{k}) = f(j,R^*)$. Since $f(j,R^*) \geq f(j,R)$, we still have that $f(j+1,R) \leq f(j+1,R^*)$.

We now show that the optimal $f(j+1,R)$ allows GPC to achieve the minimum bandwidth footprint. Assume for the sake of contradiction that there is an optimal R^* that has a lower bandwidth footprint than R. Bandwidth footprint is $\beta = B\cdot\hat{s}$, and by Lemma 5, $B \geq \hat{n}\cdot f(j,R^*)$. Then R^* using less bandwidth than R can only mean that $f(j,R^*) < f(j,R)$ for $j=\hat{n}$ — a contradiction. □