Two-Server Private Information Retrieval in Sublinear Time and Quasilinear Space

Alexandra Henzinger* Seyoon Ragavan†

July 2, 2026

Abstract

We build two-server private information retrieval (PIR) that achieves information-theoretic security and strong double-efficiency guarantees. On a database of $n > 10^6$ bits, the servers store a preprocessed data structure of size $1.5\sqrt{\log_2 n} \cdot n$ bits and then answer each PIR query by probing $12 \cdot n^{0.82}$ bits in this data structure. To our knowledge, this is the first information-theoretic PIR with any constant number of servers that has quasilinear server storage $n^{1+o(1)}$ and polynomially sublinear server time $n^{1-\Omega(1)}$.

Our work builds on the PIR-with-preprocessing protocol of Beimel, Ishai, and Malkin (CRYPTO 2000). The insight driving our improvement is a compact data structure for evaluating a multivariate polynomial and its derivatives. Our data structure and PIR protocol leverage the fact that Hasse derivatives can be efficiently computed on-the-fly by taking finite differences between the polynomial's evaluations. We further extend our techniques to improve the state-of-the-art in PIR with three or more servers, building on recent work by Ghoshal, Li, Ma, Dai, and Shi (TCC 2025).

On an 11 GB database with 1-byte records, our two-server PIR encodes the database into a 1 TB data structure—which is 4,500,000× smaller than that of prior two-server PIR-with-preprocessing schemes, while maintaining the same communication and time per query. To answer a PIR query, the servers fetch and send back 4.4 MB from this data structure, requiring 2,560× fewer memory accesses than linear-time PIR. The main limitation of our protocol is its large communication complexity, which we show how to shrink to $n^{0.31} \cdot \mathsf{poly}(\lambda)$ using compact linearly homomorphic encryption.

*ahenz@csail.mit.edu. Supported by the NSF GRFP under Grant No. 2141064, a MIT EECS Great Educators fellowship, gifts from Apple, Google, and Meta, and NSF CNS-2054869. This work was done in part while the author was visiting the Simons Institute for the Theory of Computing.

†sragavan@mit.edu. Supported by NSF CNS-2154149, a Simons Investigator Award, the Defense Advanced Research Projects Agency (DARPA) under Contract No. HR0011-25-C-0300, and Amazon Research Awards. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the Defense Advanced Research Projects Agency (DARPA). Supported in part by Jane Street. This work was done in part while the author was visiting the Simons Institute for the Theory of Computing and the Challenge Institute for Quantum Computation at UC Berkeley.

Contents

1 Introduction **1**
- 1.1 Our Results . . . 2
- 1.2 Technical Overview . . . 5
- 1.3 Related Work . . . 8

2 Background: Definitions and Framework **9**
- 2.1 PIR with Preprocessing . . . 10
- 2.2 Binary Entropy and Multinomial Coefficients . . . 11
- 2.3 Representing the Database as a Polynomial . . . 12

3 Two-Server PIR with Preprocessing **13**
- 3.1 Construction . . . 14
- 3.2 Analysis . . . 15
- 3.3 Shrinking the Communication with Homomorphic Encryption . . . 17
- 3.4 Connection to Locally Decodable Codes . . . 18

4 Background: Higher-Degree Polynomials and Hasse Derivatives **18**
- 4.1 High-Degree Database Encodings . . . 19
- 4.2 Secret Sharing . . . 19
- 4.3 Hasse Derivatives and Hermite Interpolation . . . 19

5 Many-Server PIR with Preprocessing **20**
- 5.1 Framework for Information-Theoretic PIR with Preprocessing . . . 21
- 5.2 Construction Optimizing Server Storage . . . 22
- 5.3 Construction Optimizing Server Time . . . 23

6 Evaluation **23**

7 Conclusion **26**

A Asymptotics of Count **35**

B Interpolating and Encoding the Database **41**

C Additional Material on Two-Server PIR with Preprocessing **43**
- C.1 Proof of Corollary 3.3 . . . 43
- C.2 Shrinking the Communication with Fully Homomorphic Encryption . . . 44
- C.3 Shrinking the Communication with Linearly Homomorphic Encryption . . . 45

D Additional Material on Many-Server PIR with Preprocessing **49**
- D.1 Background on Secret Sharing, Hasse Derivatives, and Hermite Interpolation . . . 50
- D.2 Additional Material on Construction Optimizing Server Storage . . . 52
- D.3 Additional Material on Construction Optimizing Server Time . . . 56

E Additional Material on Evaluation **57**

1 Introduction

Every day, users fetch data from online databases and, in the process, reveal sensitive information—e.g., the websites they browse, the places they visit, and the friends they talk to—to untrusted servers and services [Zve13]. Cryptography can eliminate this leakage: with *private information retrieval* (PIR), a user can fetch a record from a remote database, while hiding which record was fetched from the servers hosting the database [CGKS95, KO97]. PIR is at the core of systems for private web search [ASA+21, HDCGZ23, ABG+24], private media consumption [GCM+16, MW22, AAAG23], metadata-private messaging [MOT+11, AS16, CSM+20, HSW23, TWG24, KRAP25], private auditing [HHCG+23, HPW25, LHCG25, KCG21], private set intersection [LLMT25], private e-commerce and advertising [Jue01, HOG11, BKMP12, GLM16, SSHD21, ZMA22], private machine learning [LJX+24], and private storage [DFL+20, MVM25]. In recent years, PIR has made the leap from theory to real-world deployment in Apple's private caller-ID lookups [Art] and Google's Chromebook device enrollment [Yeo23].

More formally, a PIR protocol runs between a set of servers, who store an n-bit database, and a user, who holds an index $i \in \{1, 2, \ldots, n\}$. At the end of the protocol, the user learns the ith bit of the database, while the servers learn *nothing* about the index i. Chor, Goldreich, Kushilevitz, and Sudan [CGKS95] showed that PIR with information-theoretic privacy and $o(n)$ communication is possible, when relying on a non-collusion assumption between two (or more) servers. Kushilevitz and Ostrovsky [KO97] constructed PIR with $o(n)$ communication and a single server, by relaxing the privacy guarantee to be computational and relying on cryptographic hardness. In both settings, the communication complexity of modern PIR is strikingly small: it is subpolynomial in n with two servers [DG16, GKS25], or polylogarithmic in n with cryptographic assumptions and either two servers [BGI16] or a single server [KO97, CMS99, GR05, Lip05, DGI+19].

However, the privacy guarantees afforded by PIR come at a steep computational cost. While a non-private array lookup takes $O(1)$ time on a RAM machine, Beimel, Ishai, and Malkin [BIM00] proved that a PIR-lookup requires $\Omega(n)$ server time, regardless of the number of PIR servers and regardless of whether the scheme's privacy is information-theoretic or computational. More precisely, the PIR servers must touch every bit of the database when answering a query—otherwise, the server(s) would learn that the user is not reading whichever locations they did not touch. This linear server time is a painful bottleneck on databases that are many gigabytes in size and queried by millions of users, as in today's search engines, content-delivery platforms, and social networks [HDCGZ23, ABG+24, DCG23].

PIR with preprocessing. To circumvent this linear-time lower bound, Beimel, Ishai, and Malkin proposed the *PIR-with-preprocessing* model, which lets the servers run a one-time preprocessing step to produce a data structure that encodes the database [BIM00]. Then, the PIR servers use this data structure to answer PIR queries from any number of users.[1] If each location in the data structure depends on many (or all) bits of the database, the linear-server-time lower bound no longer applies: the servers could answer PIR queries by probing a sublinear number of locations, running in $o(n)$ time. Unlike other PIR models (e.g., offline/online PIR, detailed in Section 1.3), the PIR-with-preprocessing model can achieve sublinear server time even if each user makes a single query. If the database contents change, the servers must update their data structure, but no additional communication with the users is needed.

Despite decades of study, few techniques to speed up PIR using preprocessing are known, and those that exist offer limited improvements. All known PIR-with-preprocessing schemes either (1) save only polylogarithmic factors in server time [WY05, Lip09, Lee25, CIMR25], (2) require a super-constant number of non-colluding servers (i.e., growing with the database size) [LLF+25, GLM+25], or (3) encode the database

[1] This model has also been referred to as "unkeyed" doubly efficient PIR [LMW23, LMW25].

in a way that incurs large overheads in server storage. In the last category, on a database of size $n \in \mathbb{N}$, two types of schemes exist:

- Information-theoretic schemes, which use a constant number of servers, require server storage $n^{1+\Theta(1)}$ that is polynomially larger than the original database [BIM00, GLM+25]. The smaller the exponent in this polynomial, the smaller the gains in server time. In concrete terms, with two servers and on databases that are gigabytes in size, the servers would store a data structure that is *5 to 8 orders of magnitude* larger than the original database (cf. Table 1).

- Computational schemes, which use a single server, offer quasilinear server storage $n^{1+o(1)}$ and subpolynomial server time $n^{o(1)}$ [BIPW17, CHR17, LMW23]. However, these asymptotics come with prohibitive constants and factors of the security parameter: the PIR schemes either use idealized program obfuscation [BIPW17, CHR17] or directly work with a cryptographic assumption [LMW23] in a way that, on a 370 MB database, would take *128 million core-years* to build the data structure and *3 core-weeks* to answer a PIR query, whereas linear-time PIR takes 0.03 core-seconds [OPPW24].

In all cases, PIR schemes with preprocessing have never been shown to run concretely faster than linear-time PIR, on any practically relevant database size.

This paper takes another step towards speeding up PIR using preprocessing. We introduce the first information-theoretically private PIR-with-preprocessing scheme with any constant number of servers (in our case, two) that offers quasilinear server storage $n^{1+o(1)}$ and polynomially sublinear server time $n^{1-\Omega(1)}$. Concretely, on databases that are gigabytes in size, our scheme encodes the database into a data structure that is four to six orders of magnitude smaller than that of prior information-theoretic, two-server PIR with preprocessing [BIM00, GLM+25]—while matching their communication cost and their server work per query. A preliminary implementation shows that our PIR servers can answer queries at higher throughput than the fastest two-server, linear-time PIR, though our scheme requires more storage and incurs larger communication costs.

1.1 Our Results

We present new constructions of PIR with preprocessing.

Two-server PIR with preprocessing. In Section 3, we introduce a general construction of two-server PIR with preprocessing. From one instantiation of our scheme, on database size n, we obtain the first information-theoretic PIR-with-preprocessing scheme, with any constant number of servers, that has server storage $n^{1+o(1)}$ and server time $n^{1-\Omega(1)}$. We prove the following result, with no hidden constants:

Informal Theorem 1.1. *For any database of $n > 10^6$ bits, there exists an information-theoretically private two-server PIR-with-preprocessing scheme, where the database encoding stored on the servers consists of $1.5\sqrt{\log_2 n} \cdot n$ bits and the servers answer each PIR query by reading at most $12 \cdot n^{0.82}$ bits in the encoding and sending them back to the user.*

More broadly, our construction achieves the best-known tradeoff between the server storage and the server time per query among two-server PIR-with-preprocessing protocols [BIM00, GLM+25]. We give a full description of this tradeoff space, as well as a comparison to the two-server PIR-with-preprocessing protocols of Beimel, Ishai, and Malkin [BIM00] and Ghoshal, Li, Ma, Dai, and Shi [GLM+25], in Fig. 1. Another concrete instantiation of our construction gives the following scheme:

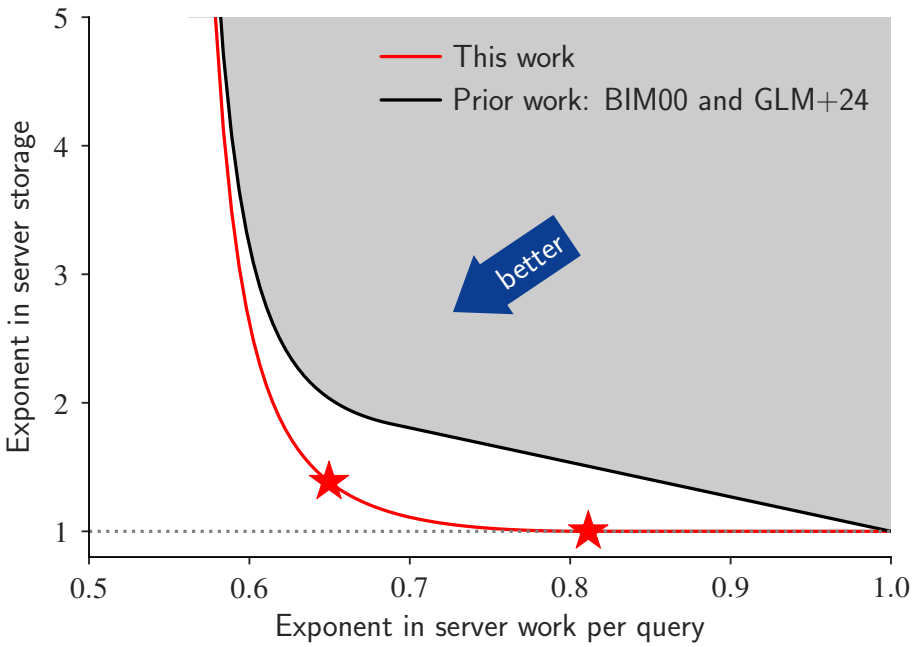

Figure 1: Asymptotic costs of two-server PIR with preprocessing: [BIM00, Theorem 4] and [GLM+25, Theorem 6.4] (which coincide in the two-server setting) and this work (Theorem 3.2). Each scheme supports a tradeoff between the server storage and the server work per query: on database size n, the storage is $\widetilde{O}(n^a)$ and the work is $\widetilde{O}(n^b)$, for constants $a, b \in \mathbb{R}^{>0}$ on the y-axis and x-axis. In each scheme, the communication is equal to the server work per query. The grey region combines prior work with standard re-balancing [CGKS95, Section 4.3]. We highlight Informal Theorems 1.1 and 1.2 with stars.

Informal Theorem 1.2. *For any sufficiently large database size $n \in \mathbb{N}$, there is an information-theoretically private two-server PIR-with-preprocessing scheme, where the database encoding stored on the servers consists of $n^{1.4}$ bits and the servers answer each PIR query by reading at most $n^{0.66}$ bits in the encoding and sending them back to the user.*

One limitation of our PIR schemes is that, in minimizing the servers' time, they incur larger per-query communication costs than linear-time PIR. Namely, in our schemes and in prior two-server, information-theoretic PIR with preprocessing [BIM00, GLM+25], the communication per query is equal to the server time per query. That is, the schemes from Informal Theorems 1.1 and 1.2 respectively have communication cost $12n^{0.82}$ and $O(n^{0.66})$ bits. We show how to shrink this communication—while retaining the same asymptotic efficiency—using cryptography, at the cost of providing only computational privacy. Using compact linearly homomorphic encryption [GM84, ElG85, Pai99, Reg09, DGI+19], we obtain PIR with preprocessing with the same server storage and server time as in Informal Theorem 1.1 (up to factors of the security parameter λ), but where the communication is reduced to $n^{0.31} \cdot \mathsf{poly}(\lambda)$. Using compact fully homomorphic encryption [RAD78, Gen09, BV11], we obtain PIR with preprocessing with the same server storage and server time (up to λ factors), but where the communication is $\mathsf{poly}(\lambda) \cdot \log n$. That is:

Informal Theorem 1.3. *Assuming compact linearly homomorphic encryption over $\mathbb{F}_2$, for any database size $n \in \mathbb{N}$, there is a computationally-private two-server PIR-with-preprocessing scheme, where the database encoding stored on the servers consists of $n^{1+o(1)}$ bits, the servers answer each query in $n^{0.82} \cdot \mathsf{poly}(\lambda)$ time, and the communication and client time per query is $n^{0.31} \cdot \mathsf{poly}(\lambda)$.*

Extending to more servers. In Sections 4 and 5, we extend our techniques to work with three or more servers. In our schemes, the number of servers is a constant—i.e., it does not grow with the size of the database—and so our schemes support databases of any size. Our schemes strictly improve on known PIR-with-preprocessing protocols with a constant number of servers [BIM00, GLM+25]. Much like in the two-server case, specific parameterizations of our schemes offer quasilinear storage and polynomially sublinear server time: with $s \geq 2$ servers, the time is $n^{\alpha(s)}$ for $\alpha(s) = 0.5 + 1/\log s + o(1/\log s)$. As in prior work [BIM00, GLM+25], our schemes adapt to the setting where a bounded number of servers may collude.

Concrete efficiency. In Section 6, we give concrete costs of our information-theoretic, two-server PIR with preprocessing. On an 11 GB database with 1-byte records, our servers encode the database into a 1 TB data structure—which is 4,500,000× smaller than the encoding produced by prior two-server schemes [BIM00, GLM+25], while achieving the same communication and time per query.

To answer a PIR query to this 11 GB database, the servers in our scheme fetch and send back 4.4 MB from their preprocessed data structure. That is, relative to the fastest two-server, linear-time PIR (with square-root communication) [CGKS95], our scheme blows up the storage by 93× and blows up the communication by 37×, while decreasing the number of memory accesses per query by 2,560×. A preliminary implementation, running on a large AWS instance, shows that this translates to 9× higher server throughput (i.e., queries answered per second, including network costs).

As mentioned above, the communication in our scheme could be significantly reduced using homomorphic encryption, without increasing the number of memory accesses (or the asymptotic server work) per query. We leave evaluating and optimizing the concrete efficiency of this direction to future work.

Abstract view of our techniques. Finally, we sketch our approach before diving into more detail in Section 1.2. All prior information-theoretic PIR-with-preprocessing schemes (with a constant number of servers and with server time $n^{1-\Omega(1)}$) follow a strategy of Beimel, Ishai, and Malkin [BIM00, WY05, GLM+25]: (1) build an "imbalanced" PIR scheme where the queries are extremely short, namely $c \cdot \log n$ bits for a constant c, and (2) have each server precompute a lookup table that stores the answer to each of the $2^{c \cdot \log n} = n^c$ possible queries. To answer a PIR query, the servers make one RAM lookup to the corresponding location in the table; letting ℓ be the bit length of the servers' answers in the "imbalanced" PIR, this requires $O(n^c \cdot \ell)$ storage and $O(\ell)$ time per query.

Our new PIR deviates from this template: we show how to eliminate a large amount of redundancy in the lookup table produced by prior PIR schemes. We design a new data structure in which, to answer a PIR query, our servers now probe $O(\ell)$ query-dependent locations, each of constant bit length (instead of one location of bit length ℓ). As a result, the PIR servers again answer queries in $O(\ell)$ time on a RAM machine, but they hold a much leaner data structure to encode the database. This improves the tradeoff between storage and time, shown in Fig. 1, and allows for *quasilinear* storage while running in polynomially-sublinear time. In addition, in the case of three or more servers, we describe a new family of imbalanced PIR schemes that improves the final construction.

Our space-efficient data structure has implications beyond PIR with preprocessing. First, it gives new and improved constructions of information-theoretic homomorphic secret sharing with preprocessing (Remark 3.6) [ILM21]. Second, Section 3.4 shows that our data structure can be viewed as a new type of locally decodable code with a relaxed smoothness property, which we call "batch smoothness." A batch-smooth locally decodable code is parameterized by a number of batches $b \in \mathbb{N}$; decoding the ith location of a message works by querying b batches of symbols in the codeword. The smoothness guarantee is that the marginal distribution of each of these b batches of symbols is independent of the index i. To our knowledge, our result gives the first batch-smooth locally decodable code with constant alphabet size, constant number

of batches, quasilinear codeword length, and polynomially sublinear number of queries.

1.2 Technical Overview

Next, we present our construction of PIR with preprocessing. We describe an information-theoretic scheme that works with any constant number of servers $s \geq 2$, focusing on the case where none of these s servers collude.

Starting point: recasting PIR as private polynomial evaluation [BIM00, BIKR02, BIK05, WY05, BV11, LMW23, GLM+25]. We begin with a step that is common to many PIR schemes: reducing PIR to the task of private polynomial evaluation. To do so, the servers represent the database $\mathsf{DB} \in \{0,1\}^n$ as a polynomial $f_{\mathsf{DB}} : \mathbb{F}_q^m \to \mathbb{F}_q$ (with a prime field size $q \geq s$, a number of variables m, and a total degree D that we will choose later) such that evaluating this polynomial on a special set of points matches the database bits. That is, there exists a public encoding function $E : \{1, 2, \ldots, n\} \to \mathbb{F}_q^m$ so that, for each index $i \in \{1, 2, \ldots, n\}$, $f_{\mathsf{DB}}(E(i)) = \mathsf{DB}_i$. For now, we impose two requirements on the polynomial f_{DB}: it must be homogeneous and multilinear. The task for a PIR user, who wishes to read the ith bit of the database, is to learn the evaluation $f_{\mathsf{DB}}(\mathbf{p})$ at the point $\mathbf{p} = E(i) \in \mathbb{F}_q^m$, while the servers learn nothing about the point $\mathbf{p}$.

Early PIR schemes solved this task using (a variant of) Shamir secret sharing [WY05]: the user samples a random line $\mathbf{L} : \mathbb{F}_q \to \mathbb{F}_q^m$ with slope $\mathbf{p}$ and sends the jth point on this line, $\mathsf{qu}_j = \mathbf{L}(j)$, as its query to server j. Server j replies with the evaluation $f_{\mathsf{DB}}(\mathsf{qu}_j)$. The user's goal is then to reconstruct the polynomial $h = f_{\mathsf{DB}} \circ \mathbf{L}$, which is the restriction of the database polynomial f_{DB} to the line $\mathbf{L}$, as the leading coefficient of this polynomial is $f_{\mathsf{DB}}(\mathbf{p}) = \mathsf{DB}_i$.[2] Crucially, this polynomial h is univariate and of total degree D. From each server's response, the user collects one piece of information about h. So, as long as the total degree D is less than the number of servers s, the user can fully recover the polynomial h via Lagrange interpolation. While beautifully simple, this protocol restricts the database polynomial f_{DB} to have constant total degree D (at most $s-1$), which pushes the number of variables m needed to encode the database to be large (at least $n^{1/(s-1)}$). The resulting PIR scheme has large upload.

Woodruff and Yehkahnin [WY05] (and, implicitly, Beimel, Ishai, and Malkin [BIM00]) showed how to use *derivatives* to trade upload and download communication in this PIR scheme. The same idea appears in the coding-theory literature under the name "multiplicity codes" [Yek12, Kop15]. In these schemes, each of the s servers sends more information to the user: namely, on query point qu_j, server j responds with the evaluation of f_{DB} and of all of its partial derivatives up to order $\lfloor D/s \rfloor$ at point qu_j, which we write as

$$\left\{ \overline{\partial}_{\mathbf{e}} f_{\mathsf{DB}}(\mathsf{qu}_j) \,:\, \mathbf{e} \in \{0,1\}^m, \|\mathbf{e}\|_1 \leq \lfloor D/s \rfloor \right\}.$$

By applying the chain rule, the user learns the evaluation of the restricted polynomial h and of its derivatives of order up to $\lfloor D/s \rfloor$: that is, the evaluations

$$h(\mathsf{qu}_j),\ h^{(1)}(\mathsf{qu}_j),\ \ldots,\ h^{(\lfloor D/s \rfloor)}(\mathsf{qu}_j).$$

Given these $\lfloor D/s \rfloor + 1$ pieces of information about the polynomial h from each server, the user can recover h via Hermite interpolation [vzGG03]. The resulting PIR schemes place no restriction on the total degree D of the database polynomial f_{DB}. On database size n, setting $D = m = \Theta(\log n)$ produces very "imbalanced"

[2] Here, we use the homogeneity of f_{DB}. Typically, Shamir sharing instead works with a random line *passing through* the point $\mathbf{p}$, which requires $q \geq s+1$. In contrast, interpolating along a random line of *slope* $\mathbf{p}$ only needs $q \geq s$; this difference affects the server storage, which grows as q^m. This type of sharing is used in PIR [GLM+25] and in data structures for fast polynomial evaluation [BKW19, BGG+24].

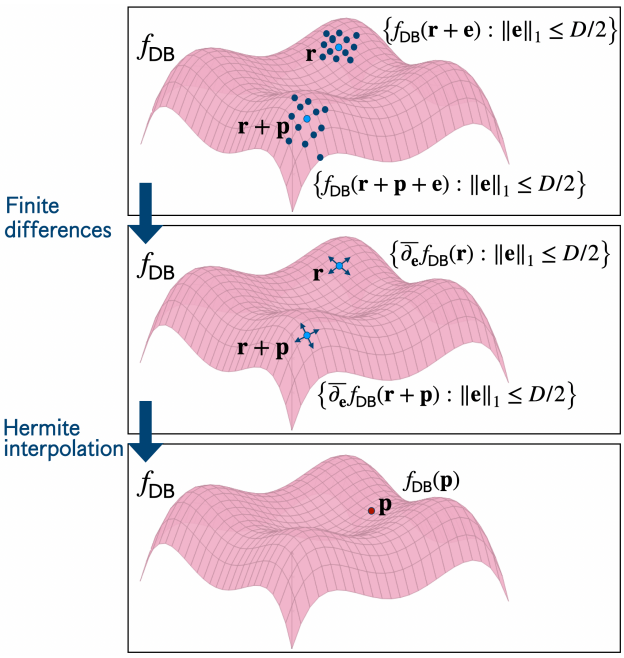

Figure 2: Our two-server PIR scheme. The client chooses an index $i \in [n]$ and finds the corresponding point $\mathbf{p} = E(i) \in \mathbb{F}_2^m$ for which $f_{\mathsf{DB}}(\mathbf{p}) = \mathsf{DB}_i$. As in prior schemes [BIM00, GLM+25], the client sends the servers additive shares of $\mathbf{p}$: server 1 gets point $\mathbf{r}$, and server 2 gets point $\mathbf{r}+\mathbf{p}$. *Top:* In our schemes, the servers send the client the evaluations of f_{DB} in a Hamming ball of radius $\lfloor D/2 \rfloor$ around the query point. To retrieve these values quickly, the servers precompute the truth table of f_{DB}. *Middle:* The client learns Hasse derivatives of f_{DB} to order $\lfloor D/2 \rfloor$ at both query points, using finite differences. *Bottom:* The client recovers $f_{\mathsf{DB}}(\mathbf{p}) = \mathsf{DB}_i$, using Hermite interpolation. In contrast, prior schemes [BIM00, GLM+25] have the servers precompute and send the Hasse derivatives to the client; i.e., they are described by the middle and bottom of the figure.

PIR: the upload from the user to each server is $m \log q = \Theta(\log n)$ bits to encode a point in $\mathbb{F}_q^m$, while the download from each server to the user is $\sum_{\ell=0}^{\lfloor D/s \rfloor} \binom{m}{\ell} = \Theta(n^\epsilon)$ bits, for some constant $1/s < \epsilon < 1$, to communicate all partial derivatives up to order $\lfloor D/s \rfloor$.

This imbalanced PIR directly gives the PIR-with-preprocessing protocol of Ghoshal, Li, Ma, Dai, and Shi [GLM+25, Section 6], which recasts and improves on the original protocol of Beimel, Ishai, and Malkin [BIM00] by viewing it in terms of Hasse derivatives. In this protocol, each of the s servers stores the truth table of f_{DB} and of all of its partial derivatives up to order $\lfloor D/s \rfloor$:

$$\mathsf{Table} = \left\{ \overline{\partial}_{\mathbf{e}} f_{\mathsf{DB}}(\mathbf{x}) \,:\, \mathbf{x} \in \mathbb{F}_q^m, \mathbf{e} \in \{0,1\}^m, \|\mathbf{e}\|_1 \le \lfloor D/s \rfloor \right\}.$$

To answer a PIR query, the servers look up and send back a record in this truth table. The server storage grows as $|\mathsf{Table}| = q^m \cdot \sum_{\ell=0}^{\lfloor D/s \rfloor} \binom{m}{\ell} = \mathsf{poly}(n)$, while the server time per query grows as $\sum_{\ell=0}^{\lfloor D/s \rfloor} \binom{m}{\ell} = \Theta(n^\epsilon)$ again for $1/s < \epsilon < 1$. Concrete choices of the constants recover [BIM00, Theorem 4] and [GLM+25, Theorem 6.4], which we visualize in Fig. 1.

Idea 1: Reconstruction via finite differences. In the previous schemes, each server stores the truth tables of f_{DB} and of all of its partial derivatives. This is highly redundant: for any linear function $g : \mathbb{F} \to \mathbb{F}$, its first-order Hasse derivative $\overline{\partial}_1 g(x)$ is simply the difference between two evaluations $g(x+1) - g(x)$. This

also holds for any *multilinear* function from $\mathbb{F}^m \to \mathbb{F}$, suggesting the following recipe: in the preprocessing stage, each server stores only the truth table of f_{DB}. To answer a PIR query, each server computes Hasse derivatives on-the-fly by taking finite differences between evaluations of f_{DB}.

Concretely, when f_{DB} is multilinear and derivatives are taken to the first order in each variable (i.e., with respect to a vector $\mathbf{e} \in \{0,1\}^m$), it holds for any query point $\mathsf{qu} \in \mathbb{F}_q^m$ that:

$$\overline{\partial}_{\mathbf{e}} f_{\mathsf{DB}}(\mathsf{qu}) = \sum_{\substack{\mathbf{e}' \in \{0,1\}^m \\ \mathbf{e}' \leq \mathbf{e}}} (-1)^{\|\mathbf{e}-\mathbf{e}'\|_1} \cdot f_{\mathsf{DB}}(\mathsf{qu} + \mathbf{e}'). \tag{1}$$

Storing only the truth table of f_{DB} shrinks the server storage down to q^m, smaller than before by a factor of $\Theta(n^\epsilon)$ for a constant $1/s < \epsilon < 1$.

To shave off lower-order overheads in server time, the servers in fact do not need to reconstruct the Hasse derivatives using Equation (1). Instead, each server can just send back the evaluations of f_{DB} in a Hamming ball around the given query point and leave this work to the client. That is, on query point $\mathsf{qu}_j \in \mathbb{F}_q^m$, server j responds with the set of evaluations

$$\left\{ f_{\mathsf{DB}}(\mathsf{qu}_j + \mathbf{e}) \,:\, \mathbf{e} \in \{0,1\}^m, \|\mathbf{e}\|_1 \leq \lfloor D/s \rfloor \right\}.$$

This scheme retains the same server time per query as before: $\sum_{\ell=0}^{\lfloor D/s \rfloor} \binom{m}{\ell} = \Theta(n^\epsilon)$ lookups on a RAM machine, for some constant $1/s < \epsilon < 1$.

In the case where the number of servers $s = 2$, the client does not need to explicitly compute Hasse derivatives, apply the chain rule, then use Hermite interpolation. Unrolling this computation yields a simple expression for the client's reconstruction that does not make any reference to derivatives. Taking $q = 2$ and D to be odd,[3] for any point $\mathbf{p} \in \mathbb{F}_2^m$ of Hamming weight D and any random mask $\mathbf{r} \in \mathbb{F}_2^m$, setting the first server's query point $\mathsf{qu}_1 := \mathbf{r}$ and the second server's query point $\mathsf{qu}_2 := \mathbf{r} + \mathbf{p}$, we see that

$$f_{\mathsf{DB}}(\mathbf{p}) = \sum_{\substack{\mathbf{e} \in \{0,1\}^m \\ \|\mathbf{e}\|_1 \leq \lfloor D/2 \rfloor \\ \mathbf{e} \leq \mathbf{p}}} \left(f_{\mathsf{DB}}(\mathsf{qu}_1 + \mathbf{e}) + f_{\mathsf{DB}}(\mathsf{qu}_2 + \mathbf{e}) \right).$$

In other words, to learn $f_{\mathsf{DB}}(\mathbf{p})$, our client sums a subset of the responses it gets from each server; which subset this is depends on the client's secret point $\mathbf{p}$.

Fig. 2 visualizes the finite differences technique in the two-server case. In Section 3, we present this construction with a self-contained, combinatorial proof that makes no reference to derivatives or Hermite interpolation. We show that it implies our new information-theoretic, two-server PIR scheme described in Fig. 1. Concrete choices of the constants yield Informal Theorems 1.1 and 1.2.

Idea 2: Going beyond multilinear polynomials. Second, we provide new storage-time tradeoffs for PIR schemes with more than two servers. To achieve this, we introduce an additional degree of freedom: the individual degree d of the database polynomial f_{DB}. Up to now, we restricted $d = 1$. Varying the individual degree d has the following effects: for a fixed total degree D and a fixed number of variables m, growing d increases the number of monomials in f_{DB}, which lets the servers represent larger databases. At the same time, as d grows, the user needs more Hasse derivatives of f_{DB} to reconstruct $f_{\mathsf{DB}}(\mathbf{p})$, as these derivatives only vanish when they are of order $> d$ in any one variable.

Previous work [BIM00, BIK05, BIKR02, GLM+25, LLF+25] targets two regimes in this tradeoff space:

[3] In Section 3, we give a more complex presentation that works for arbitrary D.

1. The $d = 1$ regime, which corresponds to a multilinear f_{DB} as described above.

2. The high-individual-degree regime, which only constrains the individual degree d of f_{DB} and sets $D = dm$. Ghoshal et al. show that this regime is beneficial when the number of servers is large [GLM+25, Theorem 4.9].

We move d freely between these two extremes. The technical challenge is to precisely pin down the database size n supported by a given set of parameters—namely, an individual degree d, a total degree D, and a number of variables m—and the number of Hasse derivatives needed. We perform this accounting in Section 4, with proofs in Appendix A.

Varying the individual degree d is compatible with our finite-differences technique: a Hasse derivative of order d can be computed on-the-fly by probing $d + 1$ locations in the truth table of f_{DB}. In Section 5, we present two protocols in this new parameter regime: first, a scheme that uses our finite differences technique; second, a scheme that precomputes Hasse derivatives as in prior work [GLM+25]. Our schemes improve on known PIR with preprocessing with a constant number of servers [BIM00, GLM+25]. Looking ahead, this improvement is visualized in Fig. 4.

Remark 1.4. When $d > 1$, the finite-differences technique does not strictly outperform the approach of precomputing and storing all Hasse derivatives. This is because taking finite differences becomes more expensive as d increases; e.g., reconstructing the first-order derivative of a univariate, degree-d polynomial using finite differences requires probing $d + 1$ evaluations of the polynomial(see Lemma D.8).

1.3 Related Work

Chor, Goldreich, Kushilevitz, and Sudan first defined and constructed private information retrieval [CGKS95]. Since then, a sequence of works improved the communication complexity of PIR in the information-theoretic [Amb97, CG97, BI01, BIKR02, Yek08, Efr12, DG16, ABL24, GKS25] and computational settings [KO97, CMS99, Cha04, Lip05, GR05, GI14, BGI15, BGI16, GH19, DGI+19]. At the same time, a second metric emerged as a point of concern for PIR: the server time per query, which Beimel, Ishai, and Malkin proved must be linear in the database size n [BIM00]. In light of this bound, a line of work sought to make linear-time PIR as concretely fast as possible [AMBFK16, ACLS18, AYA+21, GH19, MCR21, ALP+21a, MW22, HHCG+23, DPC23, ZPZS24, LMRSW24, HDCGZ23, MW24, BMW24].

Circumventing the linear-time lower bound. In addition, a number of new PIR models aimed to circumvent the linear-server-time lower bound altogether. The *batch PIR* model, introduced by Ishai, Kushilevitz, Ostrovsky, and Sahai [IKOS04], lets each user read a batch of Q database bits at once. In this model, the PIR server(s) can run in amortized sublinear time n/Q per query. However, all Q queries must be made by the same user, at the same time [IKOS04, LG15, AS16, Hen16].

The *shuffle PIR* model, also introduced by Ishai, Kushilevitz, Ostrovsky, and Sahai [IKOS06, IKLM24], shows that the PIR server(s) can run in amortized sublinear time if many users interact with the server(s) at once, anonymously. A limitation of these schemes is that their privacy hinges on the assumption that sufficiently many honest users make queries at the same time to mask each other's requests.

The *offline/online PIR* model, introduced by Patel, Persiano, and Yeo [PPY18] and Corrigan-Gibbs and Kogan [CGK20], lets each user store private, randomized, and database-dependent state. By having each user make a sequence of Q adaptively-chosen PIR queries while maintaining this local state, the server(s) again run in amortized time n/Q per query [PPY18, CGK20, KCG21, SACM21, CGHK22, LP23, ZLTS23, GZS24, ISW24, RMS24, ZPZS24, DDLO25, HPPY25, WR25]. At the same time, these schemes require every user to

make many queries for the amortization to kick in, and they necessitate communication with each user to update its local state should the database contents change.

Very recent work [LLF+25, CIMR25] proposes a new PIR model where the servers may interact with each other in a multi-round protocol to answer each query, though the schemes' privacy still relies on non-collusion. In this setting, Lazaretti et al. [LLF+25] show that three non-colluding servers can use distributed oblivious RAM [FOSZ23] to answer PIR queries in $n^{1+o(1)}$ server storage, $\mathsf{polylog}(n)$ server time, and $\Omega(\log n)$ rounds of server-server interaction per query. In contrast, all other models do not require the servers to interact to answer each PIR query.

Finally, the *PIR with preprocessing* model, introduced by Beimel, Ishai, and Malkin [BIM00], lets the PIR server(s) store additional, database-dependent state. In this model, the server(s) perform a one-time, client-independent preprocessing of the database to produce a data structure that they use to answer PIR queries. Sublinear server time is possible even if each PIR user makes only one query. If the database contents change, the server(s) must update their data structure, but no additional communication with the stateless PIR users is needed.

PIR with preprocessing. In the information-theoretic setting, with a constant number of non-colluding servers $s \geq 2$, Beimel, Ishai, and Malkin [BIM00] and Ghoshal, Li, Ma, Dai, and Shi [GLM+25] build PIR with preprocessing with $O(n^{(1+\epsilon)/s})$ server time and $\mathsf{poly}(n)$ server storage[4] for any $\epsilon > 0$. When the number of servers s grows logarithmically in n, Lazzaretti et al. [LLF+25] bring the server time down to $n^{o(1)}$ with server storage $n^{1+o(1)}$. Ghoshal et al. [GLM+25] generalize this to work with any number of servers $s = \omega(1)$ that grows with the database size. In contrast, our schemes work in the constant-server regime; they strictly improve on those of Beimel et al. [BIM00] and Ghoshal et al. [GLM+25] for any number of servers $s \geq 2$.

In the computational-privacy setting, the PIR-with-preprocessing candidates of Boyle, Pass, Ishai, and Wootters [BIPW17] and Canetti, Holmgren, and Richelson [CHR17] require virtual black-box obfuscation and new code-based assumptions. Lin, Mook, and Wichs [LMW23] show that single-server PIR-with-preprocessing is possible under the ring learning-with-errors assumption, achieving $n^{1+o(1)}$ server storage and $n^{o(1)}$ server time. However, concrete estimates [OPPW24] expect these PIR schemes to be tens of orders of magnitude slower than their linear-time counterparts. In contrast, Section 6 shows that our two-server PIR runs an order of magnitude faster than linear-time PIR on gigabyte-size databases.

As in prior work, our results combine with generic re-balancing [CGKS95, GLM+25] to improve their communication at the expense of storage or computation.

Lower bounds. Finally, Beimel, Ishai, and Malkin [BIM00] prove that, in any PIR-with-preprocessing scheme with a constant number of servers, if the servers store $b \geq 1$ extra bits of information beyond the database, they must answer PIR queries in $\Omega(n/b)$ time [BIM00]. Persiano and Yeo strengthen this bound to show that, in the single-server setting with $b = \Omega(\log n)$ extra bits, the server must run in $\Omega(n \log n/b)$ time [PY22]. Our new upper bounds are far from these lower bounds, leaving open the question of constructing—or ruling out—better information-theoretic PIR with preprocessing.

2 Background: Definitions and Framework

Notation. Let $\mathbb{N} = \{0, 1, 2, \ldots\}$ denote the set of non-negative integers. We use log to denote the base-two logarithm. For an integer $m \geq 1$, we write the set of integers $\{1, 2, \ldots, m\}$ as $[m]$. We naturally associate a binary vector $\mathbf{v} \in \{0,1\}^m$ with a set $\mathsf{supp}(\mathbf{v}) \subseteq [m]$ that contains all the positions where the vector $\mathbf{v}$ is 1.

[4] The dependence on ϵ shown by [GLM+25] is $\exp(0.368 \cdot \epsilon \cdot s^{(1+o(1))/\epsilon} \cdot \log n)$.

For a vector $\mathbf{v} \in \mathbb{N}^m$, we write its L1-norm as $\|\mathbf{v}\|_1$. For a bit string $\mathsf{msg} \in \{0,1\}^\ell$, we write $|\mathsf{msg}|$ to denote its bit length ℓ. We write $\mathbf{1}_{\mathsf{cond}}$ to denote the indicator variable that is 1 if the condition cond holds, and 0 otherwise. We write the expected value of a probability distribution $\mathcal{P}$ as $\mathbf{E}[\mathcal{P}]$. We write $\mathcal{D}_1 \equiv \mathcal{D}_2$ to denote that two distributions $\mathcal{D}_1, \mathcal{D}_2$ are identical.

Our model of computation is a RAM machine with word-size logarithmic in the input length and linear in the security parameter $\lambda \in \mathbb{N}$. We write $\mathsf{poly}(\cdot)$ to represent a fixed, unspecified polynomial. We use the standard asymptotic notation $O(f(n))$ for a function that grows smaller than $c \cdot f(n)$ for some real $c > 0$, $o(f(n))$ for a function that grows smaller than $c \cdot f(n)$ for any real $c > 0$, and $\widetilde{O}(f(n))$ for a function that grows smaller than $\mathsf{poly}(\log n) \cdot f(n)$.

2.1 PIR with Preprocessing

A PIR protocol lets a user make private reads to an n-bit database that is stored on two or more servers [CGKS95]. The servers may first encode the database into an arbitrary data structure; then, the servers use this data structure to answer queries [BIM00]. We focus on PIR schemes with a constant number of servers $s \geq 2$.

Definition 2.1 (Syntax of Private Information Retrieval with Preprocessing)**.** For any number of servers $s \geq 2$, an s-server *PIR with preprocessing* protocol is a tuple of four polynomial-time algorithms:

- $\mathsf{Preprocess}(\mathsf{DB} \in \{0,1\}^n) \to \mathsf{DB}' \in \{0,1\}^{n'}$, a deterministic algorithm that takes as input a database DB and outputs an encoded database DB'.
- $\mathsf{Query}(i \in [n]) \to (\mathsf{st}, \mathsf{qu}_1, \ldots, \mathsf{qu}_s)$, a randomized algorithm that takes as input a database index $i \in [n]$ and outputs a client-held state st and a list of s queries $\mathsf{qu}_1, \ldots, \mathsf{qu}_s$ to be sent to each of the s servers.
- $\mathsf{Answer}^{\mathsf{DB}'}(\mathsf{qu}) \to \mathsf{ans}$, a deterministic algorithm that takes as input a query qu, has oracle access to the encoded database DB', and outputs answer ans.
- $\mathsf{Reconstruct}(\mathsf{st}, \mathsf{ans}_1, \ldots, \mathsf{ans}_s) \to b \in \{0,1\}$, a deterministic algorithm that takes as input a state st and a list of s answers $\mathsf{ans}_1, \ldots, \mathsf{ans}_s$ and outputs the recovered database bit b.

We require an s-server PIR-with-preprocessing scheme to provide two properties: correctness and security, which we define next.

Definition 2.2 (Correctness of PIR with Preprocessing)**.** For every database size $n \geq 1$, for every database $\mathsf{DB} \in \{0,1\}^n$, and for every index $i \in [n]$, the client recovers the ith bit of DB by executing the protocol:

$$\Pr\left[\mathsf{Reconstruct}(\mathsf{st}, \mathsf{ans}_1, \ldots, \mathsf{ans}_s) = \mathsf{DB}_i \;\middle|\; \begin{array}{r} \mathsf{DB}' \leftarrow \mathsf{Preprocess}(\mathsf{DB}) \\ \mathsf{st}, \mathsf{qu}_1, \ldots, \mathsf{qu}_s \leftarrow \mathsf{Query}(i) \\ \mathsf{ans}_j \leftarrow \mathsf{Answer}^{\mathsf{DB}'}(\mathsf{qu}_j) \text{ for } j \in [s] \end{array}\right] = 1.$$

Definition 2.3 (Security of PIR with Preprocessing)**.** On collusion threshold $t \in [s-1]$, a PIR protocol is *information-theoretically t-private* if, for every database size $n \geq 1$ and for any two indices $i, i' \in [n]$, the PIR scheme's queries to any collection of t servers are distributed identically whether the client is reading index i or index i'. Formally, for every subset $T \subseteq [s]$ of size t,

$$\left\{ \left\{ \mathsf{qu}_j : j \in T \right\} \;\middle|\; \mathsf{st}, \mathsf{qu}_1, \ldots, \mathsf{qu}_s \leftarrow \mathsf{Query}(n, i) \right\} \equiv \left\{ \left\{ \mathsf{qu}'_j : j \in T \right\} \;\middle|\; \mathsf{st}, \mathsf{qu}'_1, \ldots, \mathsf{qu}'_s \leftarrow \mathsf{Query}(n, i') \right\}.$$

An alternate notion of PIR security exists: an s-server PIR protocol is *computationally t-private* if, on security parameter $\lambda \in \mathbb{N}$, for all polynomially-bounded database sizes $n = n(\lambda)$, pairs of indices $i, i' \in [n]$,

and subsets $T \subseteq [s]$ of size t, the above two distributions are computationally indistinguishable. When not explicitly stated, we take $t = 1$ and work in the information-theoretic setting.

Efficiency Metrics. Following prior work [BIM00, LMW23], we focus on two efficiency metrics in PIR with preprocessing: the server storage and the server time to answer a query. We define each as follows:

Definition 2.4 (Server storage)**.** On database size $n \geq 1$, the *server storage* is the worst-case bit length of the encoded database DB': i.e., $\max_{\mathsf{DB} \in \{0,1\}^n} |\mathsf{Preprocess}(\mathsf{DB})|$.

Definition 2.5 (Server time)**.** On database size $n \geq 1$ and number of servers $s \geq 2$, the *server time per query* is the expected runtime of $\mathsf{Answer}^{\mathsf{DB}'}(\cdot)$, summed over all s servers and measured in terms of the number of RAM memory accesses performed on a worst-case database.[5] That is, letting $\mathsf{MemAccess}(\mathsf{DB}', \mathsf{qu}_j)$ denote the number of bits in DB' probed by the algorithm $\mathsf{Answer}^{\mathsf{DB}'}(\mathsf{qu}_j)$, the server time (also referred to as server work) is:

$$\max_{\mathsf{DB} \in \{0,1\}^n} \mathbf{E}\left[\sum_{j \in [s]} \mathsf{MemAccess}(\mathsf{DB}', \mathsf{qu}_j) \;\middle|\; \begin{array}{r} \mathsf{DB}' \leftarrow \mathsf{Preprocess}(\mathsf{DB}) \\ \mathsf{st}, \mathsf{qu}_1, \ldots, \mathsf{qu}_s \leftarrow \mathsf{Query}(1) \end{array} \right].$$

We also report the following efficiency metrics:

Definition 2.6 (Communication)**.** On database size $n \geq 1$, the *communication* is the total bit length of all messages sent (i.e., the s queries output by Query and the s answers output by Answer) on a worst-case database.

Definition 2.7 (Server preprocessing time)**.** On database size $n \geq 1$, the *preprocessing time* is the runtime of $\mathsf{Preprocess}$ on a worst-case database.

Definition 2.8 (Client time)**.** On database size $n \geq 1$ and number of servers $s \geq 2$, the *client time per query* is the sum of the worst-case runtimes of $\mathsf{Reconstruct}(\cdot)$ and $\mathsf{Query}(\cdot)$, measured in word operations.

Remark 2.9 (Database updates). If the database changes, the servers must update their preprocessed data structure. A generic transformation given in earlier PIR work [HOWW19, CGK20, KCG21, CGHK22, LMW23], which builds on techniques from dynamic data structures [BS80, OS97], shows how to minimize the cost of this repeated preprocessing. Concretely, if ℓ of the n database bits change, it suffices to run $\mathsf{Preprocess}$ on $\log \ell$ databases of size $\widetilde{O}(\ell)$ [LMW23, Theorem 5.5 and Remark 5.1].

2.2 Binary Entropy and Multinomial Coefficients

Definition 2.10 (Binary Entropy)**.** For a real value $0 \leq p \leq 1$, we write the binary entropy of p as $H(p) = -p \log p - (1-p)\log(1-p)$. As per convention, $-p \log p$ denotes $\lim_{p \to 0} -p \log p = 0$ when $p = 0$.

Definition 2.11 (Multinomial Coefficients)**.** For $k_1, \ldots, k_r \in \mathbb{N}$, and $m \in \mathbb{N}$ such that $k_1 + \ldots + k_r = m$, the multinomial coefficient is

$$\binom{m}{k_1, k_2 \ldots, k_r} = \frac{m!}{k_1! k_2! \ldots k_r!}.$$

[5] Beimel, Ishai, and Malkin [BIM00, Definition 3] were the first to measure the server time in terms of the number of memory accesses. In their work and in all PIR schemes in this paper, the number of word operations performed by the servers is linear in the number of memory operations. By the PIR scheme's security, it suffices to measure the number of memory operations when the client is querying a fixed database entry.

As per convention, the binomial coefficient for non-negative integers m, k is

$$\binom{m}{k} = \begin{cases} \binom{m}{k,m-k}, & \text{if } k \le m, \\ 0, & \text{otherwise.} \end{cases}$$

2.3 Representing the Database as a Polynomial

As in prior work [BIM00, WY05, BIK05, DG16, GLM+25], all of our PIR schemes begin by representing the n-bit database $\mathsf{DB} \in \{0,1\}^n$ as a polynomial f_{DB}, whose evaluations on a chosen set of n points match the database bits. In more detail, each of our PIR schemes is parametrized by integers $D \ge 0$, $d \ge 0$, $m \ge 1$, and a prime field size $q \ge \max(d+1, s)$. We define the sets

$$S_{D,d,m} = \{(x_1, x_2, \ldots, x_m) \in \{0,1,\ldots,d\}^m \,:\, x_1 + \ldots + x_m = D\} \subseteq \mathbb{Z}^m$$
$$S_{\le D,d,m} = \bigcup_{d' \in \{0,1,2,\ldots,D\}} S_{d',d,m},$$

and correspondingly define $\mathsf{Count}(D,d,m) = |S_{D,d,m}|$ and $\mathsf{Count}_{\le}(D,d,m) = |S_{\le D,d,m}|$.[6] For a positive integer $i \in [\mathsf{Count}(D,d,m)]$ (respectively, $[\mathsf{Count}_{\le}(D,d,m)]$), we let $E_{D,d,m}(i)$ (respectively, $E_{\le D,d,m}(i)$) in $\mathbb{Z}^m$ be the ith element of $S_{D,d,m}$ (respectively, $S_{\le D,d,m}$) in lexicographic order. Then, in our PIR schemes:

- In Section 3 with two servers, we take $q = 2$ and $d = 1$ and encode the database DB as an m-variate polynomial $f_{\mathsf{DB}} : \mathbb{F}_q^m \to \mathbb{F}_q$ that is homogeneous with total degree D, such that

  $$\forall\, i \in [n], \quad f_{\mathsf{DB}}\left(E_{D,1,m}(i)\right) = \mathsf{DB}_i,$$

  where DB_i denotes the ith bit of the database lifted to a value in $\mathbb{F}_q$.

- In Section 5 with three or more servers, the setup is similar except (a) we vary q and d; (b) f_{DB} still has total degree $\le D$ but need not be homogeneous; and (c) we interpolate over the slightly larger set $S_{\le D,d,m}$ and require

  $$\forall\, i \in [n], \quad f_{\mathsf{DB}}\left(E_{\le D,d,m}(i)\right) = \mathsf{DB}_i.$$

Throughout this paper, we denote each of these encoding routines as the functions $\mathsf{Interpolate}(\mathsf{DB}, D, d, m, q) \to f_{\mathsf{DB}}$ and $\mathsf{Interpolate}(\mathsf{DB}, \le D, d, m, q) \to f_{\mathsf{DB}}$.

Interpolating multilinear polynomials. We next state a well-known lemma showing how to compute the polynomial f_{DB} in the special case that f_{DB} is restricted to be multilinear (i.e., $d = 1$). Later on in Section 4, we generalize these results to arbitrary polynomials f_{DB}.

Lemma 2.12. *The following all hold:*

1. Interpolating the database*: For any D and m such that $n \le \mathsf{Count}(D,1,m)$, we can take $f_{\mathsf{DB}} : \mathbb{F}_q^m \to \mathbb{F}_q$ to be the following:*

   $$f_{\mathsf{DB}}(x_1, \ldots, x_m) = \sum_{i=1}^{n} \mathsf{DB}_i \cdot \prod_{j \in \mathsf{supp}(E_{D,1,m}(i))} x_j.$$

   This f_{DB} is homogeneous and multilinear with total degree D.

[6] We often abuse notation and treat points in $S_{D,d,m}$ as points in $\mathbb{F}_q^m$; this correspondence is injective since $q > d$.

Construction 3.1 (Two-Server PIR with Preprocessing)**.** Parameterized by database size n, total degree $D = D(n) \geq 0$, and number of variables $m = m(n) \geq 1$ such that $\binom{m}{D} \geq n$. Define $E_{D,1,m}(\cdot)$ and $\mathsf{Interpolate}(\mathsf{DB}, D, 1, m, 2)$ as in Section 2.3. Define the set $\mathcal{T}_{D,m,\mathbf{p}} \subseteq S_{\leq\lfloor D/2\rfloor,1,m}$ for any $\mathbf{p} \in S_{D,1,m}$ as in Definition 3.5.

$\mathsf{Preprocess}(\mathsf{DB} \in \{0,1\}^n) \to \mathsf{Table}$

- Let $f_{\mathsf{DB}} = \mathsf{Interpolate}(\mathsf{DB}, D, 1, m, 2)$.
- Build $\mathsf{Table} : \mathbb{F}_2^m \to \mathbb{F}_2$ that, for each point $\mathbf{x} \in \mathbb{F}_2^m$, stores the evaluation $f_{\mathsf{DB}}(\mathbf{x}) \in \mathbb{F}_2$.

$\mathsf{Query}(i \in [n]) \to \left(\mathsf{st}, \mathsf{qu}_1, \mathsf{qu}_2\right)$

- Sample $\mathbf{r} \xleftarrow{R} \mathbb{F}_2^m$.
- Compute $\mathbf{p} = E_{D,1,m}(i) \in \mathbb{F}_2^m$.
- Output $\mathsf{st} = \mathbf{p}$, $\mathsf{qu}_1 = \mathbf{r}$, $\mathsf{qu}_2 = \mathbf{r} + \mathbf{p}$.

$\mathsf{Answer}^{\mathsf{Table}}(\mathsf{qu} \in \mathbb{F}_2^m) \to \mathsf{ans}$

- Initialize $\mathsf{ans} = \{\}$.
- For $\mathbf{e} \in S_{\leq\lfloor D/2\rfloor,1,m}$, append the evaluation $f_{\mathsf{DB}}(\mathsf{qu} + \mathbf{e})$ to ans, where $f_{\mathsf{DB}}(\mathsf{qu} + \mathbf{e})$ is retrieved by looking up the corresponding entry in Table.
- Output ans.

$\mathsf{Reconstruct}(\mathsf{st}, \mathsf{ans}_1, \mathsf{ans}_2) \to b$

- Parse $\mathbf{p}$ from st. Parse ans_1 as $f_{\mathsf{DB}}(\mathsf{qu}_1 + \mathbf{e})$ and ans_2 as $f_{\mathsf{DB}}(\mathsf{qu}_2 + \mathbf{e})$, for $\mathbf{e} \in S_{\leq\lfloor D/2\rfloor,1,m}$.
- Output the bit b computed as:

$$\sum_{\mathbf{t} \in \mathcal{T}_{D,m,\mathbf{p}}} f_{\mathsf{DB}}(\mathsf{qu}_1 + \mathbf{t}) + f_{\mathsf{DB}}(\mathsf{qu}_2 + \mathbf{t}).$$

Figure 3: Our two-server PIR-with-preprocessing protocol.

2. Runtime of full-domain evaluation*: For any m and multilinear $f : \mathbb{F}_2^m \to \mathbb{F}_2$, the set of evaluations $\{f(\mathbf{x}) : \mathbf{x} \in \mathbb{F}_2^m\}$ can be computed in time $O(2^m \cdot m)$.*

3. Runtime of encoding*: For any $i \in [n]$, the encoding $E_{D,1,m}(i)$ can be computed in $\mathsf{poly}(D, m)$ time.*

4. Asymptotic number of monomials*: Let $\theta \in (0,1)$ be a real constant, assume m grows, and take any $D = (\theta \pm o(1)) \cdot m$. Then, we have*

$$\log \mathsf{Count}(D, 1, m) = \log \binom{m}{D} = (H(\theta) + o(1)) \cdot m.$$

Moreover, if $\theta \leq 1/2$, then $\log \mathsf{Count}_{\leq}(D, 1, m) = \log\left(\sum_{D' \leq D} \binom{m}{D'}\right) = (H(\theta) + o(1)) \cdot m$. Otherwise, $\log \mathsf{Count}_{\leq}(D, 1, m) = (1 + o(1)) \cdot m$.

Proof Sketch. Items 1 and 4 are used in prior PIR schemes [BIM00, BIK05, GLM+25]. For self-containment, we prove Item 2 as a special case of Lemma D.6, Item 3 as a special case of Lemma B.1, and Item 4 as a special case of Theorem A.2. □

3 Two-Server PIR with Preprocessing

In this section, we describe a new two-server construction of PIR with preprocessing. We present the PIR protocol in Fig. 3; its innovation is to perform reconstruction via finite differences.

In particular, compared to the protocols of Beimel, Ishai, and Malkin [BIM00] and Ghoshal, Li, Ma, Dai, and Shi [GLM+25], our user constructs its PIR queries in the exact same way. However, in our scheme, each server sends different PIR responses back to the user: rather than replying with the evaluations of the

database polynomial and its derivatives at the given query point, each server replies with the evaluations of the database polynomial in a Hamming ball around the query point. The user then runs a new reconstruction routine to recover the database bit read. With this technique, our scheme polynomially shrinks the server storage compared to prior work, while maintaining the same server time.

In Section 3.1, we describe the protocol. In Section 3.2, we analyze its complexity. In Section 3.3, we show how to shrink its communication complexity using cryptography. In Section 3.4, we describe new implications for locally decodable codes.The main result of this section is the following theorem:

Theorem 3.2 (Information-theoretic two-server PIR with preprocessing)**.** *For any constant $0 < \theta \leq 1/2$ and database size $n \in \mathbb{N}$, there is an information-theoretically private, two-server PIR-with-preprocessing scheme with:*

- *server storage and preprocessing time $n^{1/H(\theta)+o(1)}$,*
- *server time per query $n^{H(\theta/2)/H(\theta)+o(1)}$, and*
- *communication and client time per query $n^{H(\theta/2)/H(\theta)+o(1)}$.*

The next corollaries, which are formal counterparts of Informal Theorems 1.1 and 1.2, follow by plugging concrete parameters into the construction in Fig. 3.

Corollary 3.3 (Quasilinear storage and sublinear time)**.** For any database of $n > 10^6$ bits, there exists an information-theoretically private, two-server PIR-with-preprocessing scheme with server storage $\leq 1.5 \cdot \sqrt{\log n} \cdot n$ bits, and communication and server time per query $\leq 12 \cdot n^{0.82}$.

Corollary 3.4 (Polynomial storage and sublinear time)**.** For any database size $n \in \mathbb{N}$, there exists an information-theoretically private, two-server PIR-with-preprocessing scheme with server storage and server preprocessing time $n^{1.39+o(1)}$, and communication and server time per query $n^{0.65+o(1)}$.

3.1 Construction

We introduce a useful definition, which comes up in the user's reconstruction:

Definition 3.5. For any $D, m \in \mathbb{N}$ and point $\mathbf{p} \in S_{D,1,m}$, define the collection $\mathcal{T}_{D,m,\mathbf{p}} \subseteq S_{\leq \lfloor D/2 \rfloor,1,m}$ as follows:

- If D is odd, we define $\mathcal{T}_{D,m,\mathbf{p}} = \left\{ \mathbf{t} \in S_{\leq (D-1)/2,1,m} \,:\, \mathsf{supp}(\mathbf{t}) \subseteq \mathsf{supp}(\mathbf{p}) \right\}$.
- If D is even, we take $i_{\mathbf{p}} \in [m]$ to denote an arbitrary element of $\mathsf{supp}(\mathbf{p})$ (say, its minimal element). Then, we define

$$\mathcal{T}_{D,m,\mathbf{p}} = \left\{ \mathbf{t} \in S_{\leq (D-2)/2,1,m} \,:\, \mathsf{supp}(\mathbf{t}) \subseteq \mathsf{supp}(\mathbf{p}) \right\} \cup \left\{ \mathbf{t} \in S_{D/2,1,m} \,:\, \mathsf{supp}(\mathbf{t}) \subseteq \mathsf{supp}(\mathbf{p}) \backslash \{i_{\mathbf{p}}\} \right\}.$$

With this in mind, the construction works as follows:

Preprocessing. The two servers interpolate the database into a homogeneous, multilinear polynomial f_{DB} over the field $\mathbb{F}_2$, with total degree D and m variables. Then, the servers each build the truth table of the polynomial f_{DB} over $\mathbb{F}_2^m$.

Querying. On index $i \in [n]$, the user computes the point $\mathbf{p} = E_{D,1,m}(i) \in \mathbb{F}_2^m$. Then, the user samples a random line $\mathbf{L}(t) : \mathbb{F}_2 \to \mathbb{F}_2^m$ of slope $\mathbf{p}$ and sends each server one point on this line. In other words, the user samples $\mathbf{r} \xleftarrow{R} \mathbb{F}_2^m$ and sends server 1 the point $\mathsf{qu}_1 = \mathbf{r}$ and sends server 2 the point $\mathsf{qu}_2 = \mathbf{r} + \mathbf{p}$.

Answering. For $j \in \{1, 2\}$, given query point $\mathsf{qu}_j \in \mathbb{F}_2^m$, server j responds with the evaluations of f_{DB} at the point qu_j as well as at all points of Hamming distance at most $\lfloor D/2 \rfloor$ from the point qu_j. Instead of computing these evaluations, which would require $\Omega(n)$ time, each server retrieves them from its preprocessed table.

Reconstruction. Given its point of interest $\mathbf{p} \in \mathbb{F}_2^m$ and the evaluations of the database polynomial f_{DB} in two balls of radius $\lfloor D/2 \rfloor$ around the points qu_1 and qu_2 in $\mathbb{F}_2^m$, the user computes and outputs

$$\sum_{\mathbf{t} \in \mathcal{T}_{D,m,\mathbf{p}}} \left(f(\mathsf{qu}_1 + \mathbf{t}) + f(\mathsf{qu}_2 + \mathbf{t}) \right).$$

Remark 3.6 (An alternate view: Information-theoretic homomorphic secret sharing with preprocessing). Beyond PIR, the protocol in Fig. 3 lets two servers preprocess a homogeneous, multilinear polynomial f in a way that, if f has a suitable total degree and number of variables, the servers can then evaluate f on a secret-shared point in $o(|f|)$ time. Generalizing Ishai, Lai, and Malavolta's construction of information-theoretic homomorphic secret sharing [ILM21], this gives a new protocol for homomorphic secret sharing *with preprocessing*. To evaluate a polynomial f that is not homogeneous, following prior work, it suffices to split f into homogeneous slices and process each separately [BKW19, BGG+24, GLM+25].

3.2 Analysis

We now prove that the PIR scheme of Fig. 3 is correct and secure. Throughout, we take $D = \theta \cdot m \pm O(1)$ for a parameter $0 < \theta \leq 1/2$ to be varied later. (The additive $O(1)$ leeway ensures that D is an integer.) The scheme encodes the database as a multilinear polynomial over $\mathbb{F}_2$. By Lemma 2.12, we have that $\mathsf{Count}(D, 1, m) = 2^{m \cdot (H(\theta) + o(1))}$. To make this $\geq n$, it suffices to take:

$$m = \frac{\log n}{H(\theta)} \cdot (1 + o(1)). \tag{2}$$

Security is immediate. Next, we address correctness and efficiency.

Correctness. We begin with a useful observation about the collection of vectors $\mathcal{T}_{D,m,\mathbf{p}}$ as defined in Definition 3.5:

Proposition 3.7. The collection $\left\{ \mathsf{supp}(\mathbf{t}) : \mathbf{t} \in \mathcal{T}_{D,m,\mathbf{p}} \right\}$ consists of exactly one of T and $\mathsf{supp}(\mathbf{p}) \backslash T$ for every subset $T \subseteq \mathsf{supp}(\mathbf{p})$, and no other sets. Moreover, every set in this collection has size $\leq \lfloor D/2 \rfloor$.

Proof. The collection $\left\{ \mathsf{supp}(\mathbf{t}) : \mathbf{t} \in \mathcal{T}_{D,m,\mathbf{p}} \right\}$ includes all sets of size $< D/2$ and excludes all sets of size $> D/2$. This implies the conclusion whenever $|T| \neq D/2$.

If D is even and $|T| = D/2$, T will appear in this collection if and only if $i_{\mathbf{p}} \notin T$. $\mathsf{supp}(\mathbf{p}) \backslash T$ also has size $D/2$, so it will appear in this collection if and only if $i_{\mathbf{p}} \notin \mathsf{supp}(\mathbf{p}) \backslash T \Leftrightarrow i_{\mathbf{p}} \in T$. Exactly one of these conditions always holds, so the conclusion follows. □

Correctness of our protocol follows from the next theorem:

Theorem 3.8. *Let $f : \mathbb{F}_2^m \to \mathbb{F}_2$ be any homogeneous, multilinear polynomial of total degree D. Then, for any two points $\mathbf{p} \in S_{D,1,m}$ and $\mathbf{r} \in \mathbb{F}_2^m$, we have:*

$$f(\mathbf{p}) = \sum_{\mathbf{t} \in \mathcal{T}_{D,m,\mathbf{p}}} \left(f(\mathbf{r} + \mathbf{t}) + f(\mathbf{r} + \mathbf{p} + \mathbf{t}) \right).$$

Proof. By linearity, it suffices to prove this when f is a monomial of the form $\mathbf{x} \mapsto \prod_{i \in U} x_i$ for some $U \subseteq [m]$ of size D. Note firstly that

$$f(\mathbf{p}) = 1 \Leftrightarrow U \subseteq \mathsf{supp}(\mathbf{p}) \Leftrightarrow U = \mathsf{supp}(\mathbf{p}), \tag{3}$$

where the last equivalence is because $|U| = D = |\mathsf{supp}(\mathbf{p})|$. For brevity, let $T_{\mathbf{p}}$ denote $\mathsf{supp}(\mathbf{p})$, and analogously define $T_{\mathbf{r}}$, $T_{\mathbf{t}}$. Then, we have:

$$\begin{aligned}
\sum_{\mathbf{t} \in \mathcal{T}_{D,m,\mathbf{p}}} (f(\mathbf{r}+\mathbf{t}) + f(\mathbf{r}+\mathbf{p}+\mathbf{t})) &= \sum_{\mathbf{t} \in \mathcal{T}_{D,m,\mathbf{p}}} \left(\mathbb{1}_{U \subseteq T_{\mathbf{r}} \oplus T_{\mathbf{t}}} + \mathbb{1}_{U \subseteq T_{\mathbf{r}} \oplus T_{\mathbf{p}} \oplus T_{\mathbf{t}}} \right) \\
&= \sum_{\mathbf{t} \in \mathcal{T}_{D,m,\mathbf{p}}} \left(\mathbb{1}_{U \subseteq T_{\mathbf{r}} \oplus T_{\mathbf{t}}} + \mathbb{1}_{U \subseteq T_{\mathbf{r}} \oplus (T_{\mathbf{p}} \setminus T_{\mathbf{t}})} \right) \\
&= \sum_{W \subseteq T_{\mathbf{p}}} \mathbb{1}_{U \subseteq T_{\mathbf{r}} \oplus W} \text{ (by Proposition 3.7)} \\
&= \sum_{W \subseteq T_{\mathbf{p}}} \mathbb{1}_{U \subseteq (T_{\mathbf{r}} \cap U) \oplus (W \cap U)}. \qquad (4)
\end{aligned}$$

We complete the proof with some casework:

- Suppose there exists an element $i \in T_{\mathbf{p}} \setminus U$. Then, we have $W \cap U = (W \oplus \{i\}) \cap U$. Pairing up subsets of $T_{\mathbf{p}}$ in this way implies that Equation (4) is 0 over $\mathbb{F}_2$. In this case, as $U \neq T_{\mathbf{p}}$, it follows that Equation (3) is also 0.
- Otherwise, we must have $T_{\mathbf{p}} \subseteq U \Rightarrow T_{\mathbf{p}} = U$, which implies that Equation (3) is 1. In Equation (4), W now ranges over all subsets of U, and therefore $(T_{\mathbf{r}} \cap U) \oplus W$ also ranges over all subsets of U. Exactly one of these subsets contains U (namely, U itself), so it follows that exactly one of the terms being summed will be 1. Thus, Equation (4) is also 1. □

Efficiency. We address each metric in sequence:

- *Server storage and preprocessing time:* the server storage is 2^m. The preprocessing consists of (1) writing out the coefficients of f_{DB}, done in $O(2^m)$ time (Lemma 2.12, Item 1), and (2) precomputing all evaluations of f_{DB}, done in $O(2^m \cdot m)$ time (Lemma 2.12, Item 2). Overall, the server storage and preprocessing time are both $\boxed{2^{m(1+o(1))} = n^{1/H(\theta)+o(1)}}$.
- *Server time per query:* the server performs $\mathsf{Count}_{\leq}(\lfloor D/2 \rfloor, 1, m)$ memory accesses. By Lemma 2.12, Item 4, this is $\boxed{2^{m(H(\theta/2)+o(1))} = n^{H(\theta/2)/H(\theta)+o(1)}}$.
- *Client time per query:* to read index $i \in [n]$, the client computes $E_{D,1,m}(i)$, done in $\mathsf{poly}(D, m) = \mathsf{poly}(\log n)$ time (Lemma 2.12, Item 3). Reconstruction takes time $\boxed{\mathsf{Count}_{\leq}(\lfloor D/2 \rfloor, 1, m) \cdot O(m) = n^{H(\theta/2)/H(\theta)+o(1)}}$.

This completes the proof of Theorem 3.2.

Corollary 3.4 follows by setting $\theta = 1/5$ in Theorem 3.2 and the above analysis. We prove Corollary 3.3 in Appendix C.1: the high-level idea is to use our construction with m variables and total degree $D = \lfloor m/2 \rfloor$ so that we can encode a database of size $\binom{m}{\lfloor m/2 \rfloor} \approx 2^m / \sqrt{m\pi/2}$ with a data structure of size 2^m. To accommodate other database sizes without paying constant overheads in server storage, we follow a classic technique of breaking the database into chunks of exponentially-decreasing size and applying the PIR scheme to each chunk. Then, we perform a more precise accounting of the parameters of the construction.

3.3 Shrinking the Communication with Homomorphic Encryption

Next, we combine our new information-theoretic PIR scheme with standard cryptographic primitives to reduce its communication cost.

Background: Homomorphic encryption. An encryption scheme is homomorphic [RAD78, Gen09] if it supports computations on encrypted data. Linearly homomorphic encryption can compute all polynomial-size linear functions on encrypted data. Fully homomorphic encryption can compute all polynomial-size circuits on encrypted data. We follow standard definitions of symmetric-key linearly and fully homomorphic encryption, which require the encryption scheme to be correct, semantically secure, and strongly compact [Hal17]. We use Halevi's notion of "multi-hop" homomorphic encryption [Hal17], which has the property that evaluating a circuit with C gates takes time $C \cdot \mathsf{poly}(\lambda)$ on security parameter λ. Classic linearly and fully homomorphic encryption schemes are strongly compact and multi-hop [GM84, ElG85, Pai99, DJ01, Gen09, BV11, BGV12, FV12, GSW13].

Composing PIR with homomorphic encryption. To shrink the communication in the PIR scheme from Fig. 3, we let the user send each server its original PIR query, along with an encryption of its secret point $\mathbf{p} \in \mathbb{F}_2^m$, using linearly or fully homomorphic encryption. Each server answers the PIR query as before, then runs the user's reconstruction under the hood of the encryption scheme. With linearly homomorphic encryption, each server first writes the user's reconstruction as a degree-two polynomial (in the secret point $\mathbf{p}$), which the server partially evaluates. Following an observation by [BIM04, Theorem 4.7], we can further shrink communication by noticing that the user only needs to read some of the bits of each server's answer, hence they can retrieve each of these bits using a succinct, computationally-private, single-server PIR protocol [KO97].[7] We give the full schemes in Appendices C.2 and C.3, proving:

Theorem 3.9 (Computational two-server PIR with preprocessing)**.** *For any constant $0 < \theta \leq 1/2$, database size $n \in \mathbb{N}$, and security parameter $\lambda \in \mathbb{N}$,*

- *assuming strongly-compact, multi-hop linearly homomorphic encryption over $\mathbb{F}_2$, for any $\epsilon > 0$ there is a computationally-private, two-server PIR-with-preprocessing scheme with*

$$n^{\frac{\theta \cdot H(\theta/2)}{H(\theta)(\theta + H(\theta/2))} + \epsilon} \cdot \mathsf{poly}(\lambda)$$

  communication and client time per query.

- *assuming strongly-compact, multi-hop fully homomorphic encryption over $\mathbb{F}_2$, there is a computationally-private, two-server PIR-with-preprocessing scheme with $\mathsf{poly}(\lambda) \cdot \log n$ communication and client time per query.*

In both cases, the server storage and preprocessing time are $n^{1/H(\theta)+o(1)}$, and the server time per query is $n^{H(\theta/2)/H(\theta)+o(1)} \cdot \mathsf{poly}(\lambda)$.

In the case of linearly homomorphic encryption, as θ approaches zero, this recovers the result of [BIM04, Theorem 4.7], which achieves $n^{\epsilon} \cdot \mathsf{poly}(\lambda)$ communication and $n^{1/2+\epsilon}$ server work per query, while retaining our improvement by a multiplicative factor of $n^{H(\theta/2)/H(\theta)+o(1)} = n^{1/2+\epsilon}$ in server storage. We additionally highlight the case of $\theta = 1/2$ below:

Corollary 3.10. Assuming strongly-compact, multi-hop linearly homomorphic encryption over $\mathbb{F}_2$, there is a computationally-private, two-server PIR-with-preprocessing scheme with server storage and preprocessing time $n^{1+o(1)}$, server time per query $n^{0.82} \cdot \mathsf{poly}(\lambda)$, and communication and client time per query $n^{0.31} \cdot \mathsf{poly}(\lambda)$.

[7] We thank Yuval Ishai for pointing us to the communication-efficient PIR schemes in [BIM04, Theorem 4.7].

3.4 Connection to Locally Decodable Codes

Finally, we translate our new PIR scheme into the language of *locally decodable codes* (LDCs) [KT00, Yek12]. We recall the standard notion of smooth LDCs:

Definition 3.11. A q-query LDC consists of three polynomial-time algorithms:

- A deterministic function $\mathsf{Encode} : \{0,1\}^n \to \Sigma^{\ell(n)}$;
- A randomized function $\mathsf{Query} : [n] \to [\ell(n)]^q$; and
- A deterministic function $\mathsf{Decode} : [n] \times [\ell(n)]^q \times \Sigma^q \to \{0,1\}$.

Correctness requires that for any $\mathbf{x} \in \{0,1\}^n$ and $i \in [n]$ we have:

$$\Pr\left[\mathsf{Decode}(i, \mathsf{qu}_1, \ldots, \mathsf{qu}_q, c_{\mathsf{qu}_1}, \ldots, c_{\mathsf{qu}_q}) = x_i \;\middle|\; \begin{array}{r} \mathbf{c} \leftarrow \mathsf{Encode}(\mathbf{x}) \\ \mathsf{qu}_1, \ldots, \mathsf{qu}_q \leftarrow \mathsf{Query}(i) \end{array}\right] = 1.$$

We say an LDC is *smooth* if, for any $i \in [n]$ and $j \in [q]$, the marginal distribution of each query qu_j (where the queries are sampled from $\mathsf{Query}(i)$) is uniformly random over $[\ell(n)]$.

To capture our new schemes, we define a relaxed smoothness property for LDCs:

Definition 3.12 (Batch-Smooth Locally Decodable Code)**.** We say a q-query locally-decodable code is *b-batch-smooth* if q is divisible by b and, for any $i, i' \in [n]$ and $j \in [b]$, the joint distribution of $\{\mathsf{qu}_{(j-1)q/b+1}, \mathsf{qu}_{(j-1)q/b+2}, \ldots, \mathsf{qu}_{jq/b}\}$ is identical to the joint distribution of $\{\mathsf{qu}'_{(j-1)q/b+1}, \mathsf{qu}'_{(j-1)q/b+2}, \ldots, \mathsf{qu}'_{jq/b}\}$ (where the queries $\mathsf{qu}, \mathsf{qu}'$ are sampled from $\mathsf{Query}(i), \mathsf{Query}(i')$ respectively).

Remark 3.13. Any smooth q-query LDC with output alphabet Σ and codeword length $\ell(n)$ can be transformed into a q-batch-smooth LDC by writing each symbol in Σ as a string of $\log|\Sigma|$ bits. The resulting q-batch-smooth LDC requires $q \log|\Sigma|$ queries and has output alphabet $\{0,1\}$ and codeword length $\ell(n)\log|\Sigma|$. This transformation is completely generic and applicable to LDCs beyond those considered in our work, including matching vector codes [Yek12, DG16].

The two-server PIR-with-preprocessing protocol of Beimel et al. [BIM00] can be viewed in either of the following ways, for a parameter $0 < \theta \le 1/2$:

- A two-query smooth LDC with codeword length $\ell(n) = n^{1/H(\theta)+o(1)}$ but with a huge alphabet $\log|\Sigma| = n^{H(\theta/2)/H(\theta)+o(1)}$; or
- A two-batch-smooth LDC with alphabet $\Sigma = \{0,1\}$, codeword length $\ell(n) = n^{(1+H(\theta/2))/H(\theta)+o(1)}$, and number of queries $q = n^{H(\theta/2)/H(\theta)+o(1)}$. (This can be seen by applying the transformation in Remark 3.13 to the previous bullet.)

Our new PIR matches the result in the first bullet, and strictly improves on the second bullet: all parameters are the same except that our codeword length is only $n^{1/H(\theta)+o(1)}$. By setting $\theta = 1/2$, we obtain the first batch-smooth LDC with constant alphabet size, constant number of batches b, codeword length quasilinear in n, and polynomially sublinear number of queries (i.e., $q = n^{1-\Omega(1)}$).

4 Background: Higher-Degree Polynomials and Hasse Derivatives

Next, we generalize the tools used to construct two-server PIR with preprocessing. Looking ahead, Section 5 uses these generalizations to build PIR with preprocessing with more servers and larger collusion thresholds.

4.1 High-Degree Database Encodings

First, we discuss how to represent a database $\mathsf{DB} \in \{0,1\}^n$ as a polynomial f_{DB} with individual degree larger than one. Recall from Section 2.3 that our PIR servers encode the database DB as a m-variate polynomial $f_{\mathsf{DB}} : \mathbb{F}_q^m \to \mathbb{F}_q$ with total degree $\leq D$, individual degree d, and a prime modulus $q \geq d+1$, such that, for every index $i \in [n]$, $f_{\mathsf{DB}}\left(E_{\leq D,d,m}(i)\right) = \mathsf{DB}_i$. For this to be possible, the parameters must give rise to a sufficiently expressive class of polynomials: we need $\mathsf{Count}_{\leq}(D,d,m) \geq n$, for the function Count defined in Section 2.3 (that counts the number of monomials in a polynomial with the given parameters).

For a given real number $\theta > 0$ and individual degree $d \geq 1$, we perform a fine-grained accounting of the asymptotic behavior of Count in Appendix A. Our results precisely characterize functions $\mathsf{CntExp}(\theta,d)$ and $\mathsf{CntExp}_{\leq}(\theta,d)$ for real number $\theta > 0$ and constant d, such that $\mathsf{Count}(\theta m,d,m) \approx 2^{m\cdot\mathsf{CntExp}(\theta,d)}$ and $\mathsf{Count}_{\leq}(\theta m,d,m) \approx 2^{m\cdot\mathsf{CntExp}_{\leq}(\theta,d)}$.

In Appendix B, we describe the algorithms needed to efficiently work with the representation of the database as a polynomial:

1. For any index $i \in [n]$, the servers and users can efficiently compute the encoding $E_{\leq D,d,m}(i)$. Lemma B.1 shows how to do so in $\mathsf{poly}(D,d,m)$ time.

2. The servers can efficiently construct the polynomial f_{DB} from the database DB. Theorem B.3 shows how to do so in $\mathsf{Count}_{\leq}(D,d,m)\cdot\mathsf{poly}(D,m,\log|\mathbb{F}|)$ time.

4.2 Secret Sharing

Next, we recall a generalization of Section 3's secret-sharing scheme to handle a bounded number of colluding servers. As in prior works [BKW19, BGKM22, BGG+24, GLM+25], we share a secret point $\mathbf{p} \in \mathbb{F}_q^m$ across s servers, of which at most t may collude, by sampling a random curve in $\mathbb{F}_q \to \mathbb{F}_q^m$ of degree t, whose leading coefficient is the secret point $\mathbf{p}$. Then, we let each share be a point on this curve. The only requirement is that $q \geq s$. (In contrast, classical Shamir sharing encodes the secret as the constant coefficient of the polynomial; this requires $q > s$, increasing the field size and thus also the storage in our preprocessing setting.)

We prove in Appendix D.1that this secret sharing is secure. Bhargava et al. [BGG+24] establish the following correctness guarantee:

Lemma 4.1 (Section 4.3 of [BGG+24])**.** *Let $\mathbf{L} : \mathbb{F} \to \mathbb{F}^m$ be a curve of degree t with leading coefficient $\mathbf{s}$. Let $f : \mathbb{F}^m \to \mathbb{F}$ be a homogeneous polynomial of total degree D. Then, the coefficient of X^{Dt} in $f \circ \mathbf{L}$ is equal to the evaluation $f(\mathbf{s})$.*

4.3 Hasse Derivatives and Hermite Interpolation

Finally, we give background on derivatives and interpolation. At the highest level, over a field of characteristic zero, Lagrange interpolation recovers a univariate polynomial f of degree D from its evaluations at $D+1$ distinct points. Hermite interpolation recovers the polynomial f from its evaluations at *fewer* than $D+1$ points, given additional evaluations of its derivatives [vzGG03, Section 5.6]. When the field has finite characteristic, working with Hasse derivatives prevents the derivatives from vanishing and enables Hermite interpolation.

Let $\mathbb{F}$ be an arbitrary field. For a polynomial $f \in \mathbb{F}[X_1,\ldots,X_m]$ and a vector $\mathbf{e} = (e_1,\ldots,e_m) \in \mathbb{N}^m$, we write $\mathsf{Coeff}_{\mathbf{e}}(f) \in \mathbb{F}$ to denote the coefficient in front of the monomial $\mathbf{X}^{\mathbf{e}} = \prod_{i=1}^m X_i^{e_i}$ in the polynomial f.

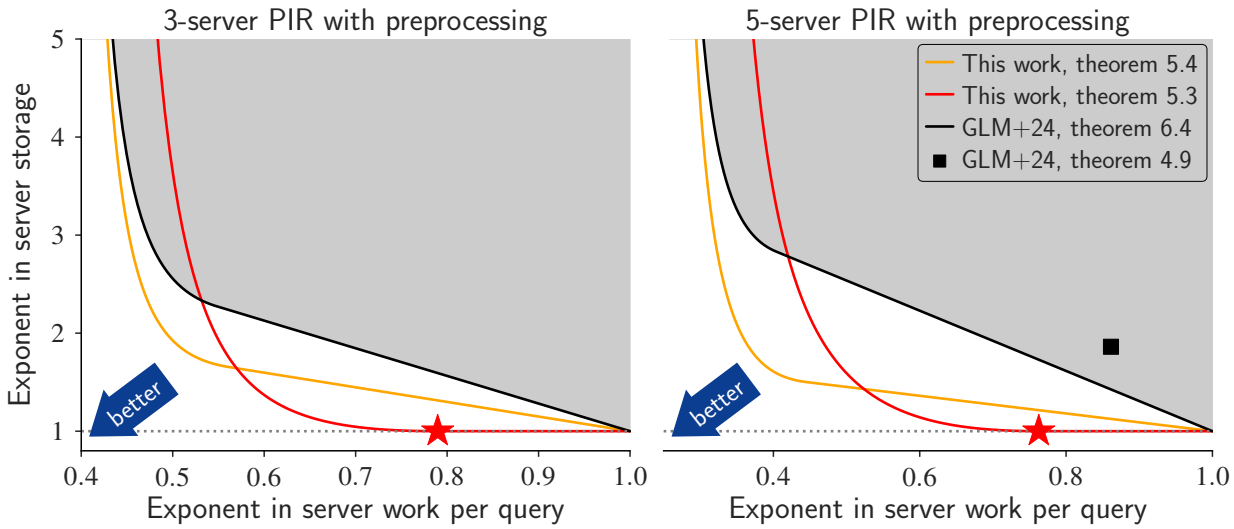

Figure 4: Asymptotic costs for many-server, information-theoretic PIR with preprocessing: [GLM+25, Theorems 4.9 and 6.4], and this work. The red and orange curves are obtained by setting $q = s, d = s-1, t = 1$ in Theorems 5.3 and 5.4 respectively, and numerically approximating $\mathsf{CntExp}(\theta, d)$ as defined in Theorem A.2. Each scheme supports a tradeoff between the server storage and the server work per query: on database size n, the storage is $\widetilde{O}(n^a)$ and the work is $\widetilde{O}(n^b)$, for constants $a, b \in \mathbb{R}^{>0}$ on the y-axis and x-axis respectively. We highlight the most efficient scheme with quasilinear server storage (Corollary D.11)with a star; for those schemes, the storage is in fact $O(n \log n)$. The grey region combines prior work with standard re-balancing [CGKS95, Section 4.3].

Definition 4.2 (Hasse Derivatives)**.** The *Hasse derivative* of a polynomial $f \in \mathbb{F}[X_1, \ldots, X_m]$ with respect to a vector $\mathbf{e} \in \mathbb{N}^m$ is defined as:

$$\overline{\partial}_{\mathbf{e}} f(X_1, \ldots, X_m) = \sum_{\substack{\mathbf{a} \in \mathbb{N}^m \\ \mathbf{a} \geq \mathbf{e}}} \left(\prod_{i=1}^{m} \binom{a_i}{e_i} \cdot \mathsf{Coeff}_{\mathbf{a}}(f) \cdot \mathbf{X}^{\mathbf{a}-\mathbf{e}} \right). \tag{5}$$

We discuss properties of Hasse derivatives in Appendix D.1. In addition, we provide well-known lemmas that allow for:

1. recovering a univariate polynomial f from evaluations of f and of its Hasse derivatives, via Hermite interpolation(see Lemma D.4),
2. recovering the restriction of a multivariate polynomial f to a univariate curve from evaluations of f and of its Hasse derivatives along the curve, via the chain rule(see Lemma D.5), and
3. efficiently evaluating the truth tables of a multivariate polynomial f and some of its Hasse derivatives, via fast polynomial evaluation(see Lemma D.6).

5 Many-Server PIR with Preprocessing

In this section, we describe new constructions of PIR with preprocessing for three or more servers. Our schemes depart from prior work [BIM00, GLM+25] in two ways:

1. We observe that Hasse derivatives can be efficiently computed on-the-fly from the truth table of the database polynomial f_{DB} by taking finite differences. In constrast, prior schemes require the servers to store the truth table of f_{DB} and of each Hasse derivative [BIM00, GLM+25].

2. We take the individual degree d of the database polynomial f_{DB} to be a tunable constant larger than one.

We provide a general, information-theoretic PIR-with-preprocessing framework that abstracts prior work in Section 5.1. Then, using the two above techniques, we give concrete PIR schemes that instantiate this framework: in Section 5.2, we present a construction that uses both techniques, tailored towards optimizing the server storage. In Section 5.3, we present a construction that uses only the second technique, tailored towards optimizing the server time per query.

Both schemes improve on the storage-time tradeoff achieved by Ghoshal et al. [GLM+25], as shown in Fig. 4. Our new construction using finite differences is the first information-theoretic, constant-server PIR-with-preprocessing scheme to achieve *quasilinear* server storage $n^{1+o(1)}$, while providing polynomially sublinear server time $n^{1-\Omega(1)}$(see Corollary D.11).

5.1 Framework for Information-Theoretic PIR with Preprocessing

Let the number of servers $s \geq 2$ be a constant and let $t \in [s-1]$ denote the collusion threshold. We again work with a prime field size $q \geq s$ and integers D, d, and $m \geq 1$ that determine the total degree, individual degree, and number of variables of the database polynomial f_{DB}. In the rest of this section, we regard $d \in [q-1]$ as a constant and take $D = \theta \cdot m \pm O(1)$ for a constant $0 < \theta \leq d/2$ to be varied later. For an integer $\ell \in \{0, 1, \ldots, D\}$, we define $\mathsf{HW}_{\ell,d,m} \subseteq \mathbb{F}_q^m$ to be the set of vectors in $\{0, 1, \ldots, d\}^m$ of Hamming weight at most ℓ.

Our PIR schemes adhere to the following template, which abstracts and generalizes the earlier schemes of Beimel et al. [BIM00] and Ghoshal et al. [GLM+25]:

- *Preprocessing*: the servers encode the database as a polynomial $f_{\mathsf{DB}} : \mathbb{F}_q^m \to \mathbb{F}_q$ of total degree D and individual degree d(see Section 4.1). The servers split this polynomial into D homogeneous slices $f_0, f_1, \ldots, f_D : \mathbb{F}_q^m \to \mathbb{F}_q$ such that $f_{\mathsf{DB}} = \sum_{\ell=0}^{D} f_\ell$ and each polynomial f_ℓ has degree ℓ. That is, each polynomial f_ℓ consists of all terms in f_{DB} that have degree ℓ. For each polynomial f_ℓ, the servers store a data structure that allows for fast evaluation of f_ℓ and of its Hasse derivatives up to order $\lfloor \ell t/s \rfloor$ at any point in $\mathbb{F}_q^m$.

- *Querying*: on index $i \in [n]$, the user computes the point $\mathbf{p} = E_{\leq D,d,m}(i) \in \mathbb{F}_q^m$ and shares it across the s servers using a secret-sharing scheme with threshold t(see Section 4.2). Formally, the user samples $\mathbf{c}_0, \ldots, \mathbf{c}_{t-1} \xleftarrow{R} \mathbb{F}_q^m$ and builds a degree-t curve $\mathbf{L} : \mathbb{F}_q \to \mathbb{F}_q^m$, whose leading coefficient is the point $\mathbf{p}$:

$$\mathbf{L}(X) = \mathbf{p} \cdot X^t + \sum_{i=0}^{t-1} \mathbf{c}_i \cdot X^i.$$

  The user sends the jth point on the curve, $\mathsf{qu}_j := \mathbf{L}(j)$, to server j.

- *Answering*: given query point $\mathsf{qu}_j \in \mathbb{F}_q^m$, for each $\ell \in \{0, 1, \ldots, D\}$, server j replies with the contents of the cells that it must probe in its data structure to recover (1) the evaluation of the polynomial f_ℓ and (2) the evaluation of all of its Hasse derivatives up to order $\lfloor \ell t/s \rfloor$, at the point qu_j.

- *Reconstruction*: given each server's response, the user runs the data structure's reconstruction routine to learn the set of evaluations $\overline{\partial}_{\mathbf{e}} f_\ell(\mathsf{qu}_j)$ for each $\ell \in \{0, 1, \ldots, D\}$, $j \in [s]$, and $\mathbf{e} \in \{0, 1, \ldots, d\}^m$ such that $\|\mathbf{e}\|_1 \leq \lfloor \ell t/s \rfloor$. Via Hermite interpolation and the chain rule(see Section 4.3), the user recovers the set of evaluations $f_\ell(\mathbf{p})$ for $\ell \in \{0, 1, \ldots, D\}$. Finally, the user computes its final output as $\mathsf{DB}_i = f_{\mathsf{DB}}(\mathbf{p}) = \sum_{\ell=0}^{D} f_\ell(\mathbf{p})$.

Next, we give two concrete PIR schemes that differ in a key component: the data structure used for fast evaluation of a polynomial and its Hasse derivatives.

5.2 Construction Optimizing Server Storage

We first formalize the core technique in our construction: performing interpolation via finite differences. In Appendix D.2, we provide an algorithm that, given evaluations of a multivariate polynomial $f : \mathbb{F}_q[X_1, \ldots, X_m] \to \mathbb{F}_q$ in a Hamming ball around any evaluation point $\mathbf{x} \in \mathbb{F}_q^m$, computes the evaluations of f and of its Hasse derivatives at point $\mathbf{x}$. The larger the Hamming ball, the higher the order of the Hasse derivatives that can be recovered. More formally:

Lemma 5.1 (Finite differences)**.** *For any prime q, multivariate polynomial $f : \mathbb{F}_q[X_1, \ldots, X_m] \to \mathbb{F}_q$ of individual degree $d \in [q-1]$, and $\ell \in \{0, 1, \ldots, dm\}$, there exists an algorithm that, for any evaluation point $\mathbf{x} \in \mathbb{F}_q^m$:*

1. *takes as input the evaluations $\left\{ f(\mathbf{x} + \mathbf{r}) : \mathbf{r} \in \mathsf{HW}_{\min(\ell,m),d,m} \right\}$, and*
2. *produces as output the evaluations $\left\{ \overline{\partial}_{\mathbf{e}} f(\mathbf{x}) : \mathbf{e} \in S_{\leq \ell,d,m} \right\}$.*

This algorithm is linear over $\mathbb{F}_q$ and does not take the point $\mathbf{x}$ itself as input. Its runtime is $\left|\mathsf{HW}_{\min(\ell,m),d,m}\right| \cdot m \cdot \mathsf{poly}(d, \log q)$.

Using this algorithm, we can think of the truth table of a multivariate polynomial f as a "data structure" for fast evaluation of f and of its Hasse derivatives at any evaluation point. More precisely:

Corollary 5.2 (Data structure for fast evaluation of a polynomial and its Hasse derivatives)**.** For any prime q, polynomial $f : \mathbb{F}_q[X_1, \ldots, X_m] \to \mathbb{F}_q$ of individual degree $d \in [q-1]$, and $\ell \in \{0, 1, \ldots, dm\}$, there exists a data structure of size $q^m \cdot \lceil \log q \rceil$ bits and computable in time $q^m \cdot \mathsf{poly}(d, m, \log q)$, such that, for any point $\mathbf{x} \in \mathbb{F}_q^m$, it is possible to compute the evaluations $\left\{ \overline{\partial}_{\mathbf{e}} f(\mathbf{x}) : \mathbf{e} \in S_{\leq \ell,d,m} \right\}$ by making $\left|\mathsf{HW}_{\min(\ell,m),d,m}\right| \cdot \lceil \log q \rceil$ RAM accesses to the data structure and running in $\left|\mathsf{HW}_{\min(\ell,m),d,m}\right| \cdot m \cdot \mathsf{poly}(d, \log q)$ time.

Finally, by plugging the data structure from Corollary 5.2 into the PIR framework of Section 5.1, we obtain a new s-server, t-private, information-theoretic PIR-with-preprocessing scheme. When setting $s = 2$ and $q = 2$, our construction recovers the PIR scheme of Section 3. More broadly, we present a construction in Appendix D.2 that proves:

Theorem 5.3 (Information-theoretic many-server PIR with preprocessing)**.** *With $s \geq 2$ servers and collusion threshold $t \in [s-1]$, for any database size $n \in \mathbb{N}$, prime $q \geq s$, constant $d \in [q-1]$, and real $\theta \in (0, d)$ such that $\theta \cdot t/s < d/(d+1)$, there is an s-server, information-theoretically t-private PIR-with-preprocessing scheme with server storage and preprocessing time $n^{\alpha+o(1)}$ and communication, server time, and client time per query $n^{\beta+o(1)}$, for:*

$$\alpha = \frac{\log q}{\mathsf{CntExp}_{\leq}(\theta, d)} \qquad \text{and} \qquad \beta = \frac{s \cdot H(\theta t/s) + \theta t \log d}{s \cdot \mathsf{CntExp}_{\leq}(\theta, d)}.$$

Size n (GB)	**Deg** D	**Vars** m	**Storage** (TB) Fig. 3	**Storage** (TB) Prior work
Databases with 1-byte records.				
122	19	40	1	1.6×10^5
82	17	40	1	1.4×10^6
37	15	40	1	4.9×10^6
11	13	40	1	4.5×10^6
2	11	40	1	7.6×10^5
Databases with 32-byte records.				
135	17	35	1	1.4×10^4
96	15	35	1	1.3×10^5
43	13	35	1	4.5×10^5
12	11	35	1	3.5×10^5
2	9	35	1	5.9×10^4

Table 1: Server storage in our two-server PIR with preprocessing and prior schemes: [BIM00, Theorem 4] and [GLM+25, Theorem 6.4], which coincide in the two-server case. In our comparisons, we allow the prior schemes to make use of rebalancing while keeping the download the same as or smaller than our corresponding scheme. Each of our schemes encodes the database as a polynomial over $\mathbb{F}_2$ with total degree D and m variables; they have the same upload, download, and server time per query. On record length ℓ bytes, the supported database size is $\binom{m}{D} \cdot \ell$ bytes (column 1); our storage is $2^m \cdot \ell$ bytes (column 4).

In Appendix D.2, we highlight special cases of the theorem that show a strict improvement over prior work [GLM+25] and that achieve quasilinear storage, as in Fig. 4. In the quasilinear-storage setting with collusion threshold $t = 1$, the larger the number of servers s, the closer the server time is to $\sqrt{n}$ (see Corollary D.11).

5.3 Construction Optimizing Server Time

Finally, we provide a construction that further shrinks the server time. Instead of using finite differences, we follow prior work [BIM00, GLM+25] and explicitly have each server store the truth table of each polynomial $f_0, f_1, \ldots, f_D$ and of all of its Hasse derivatives. This improves the server time per query, since each server no longer probes multiple locations in the truth table to back out each derivative. At the same time, this increases the server storage.In Appendix D, we prove:

Theorem 5.4 (Information-theoretic many-server PIR with preprocessing). *With $s \geq 2$ servers and collusion threshold $t \in [s-1]$, for any database size $n \in \mathbb{N}$, prime $q \geq s$, constant $d \in [q-1]$, and positive real $\theta \in (0, d)$ with $\theta \cdot t/s < d/2$, there is an s-server, information-theoretically t-private PIR-with-preprocessing scheme with server storage and preprocessing time $n^{\alpha+o(1)}$ and communication, server time, and client time per query $n^{\beta+o(1)}$, for*

$$\alpha = \frac{\log q + \mathsf{CntExp}(\theta t/s, d)}{\mathsf{CntExp}_{\leq}(\theta, d)} \qquad \text{and} \qquad \beta = \frac{\mathsf{CntExp}(\theta t/s, d)}{\mathsf{CntExp}_{\leq}(\theta, d)}.$$

6 Evaluation

We implement our information-theoretic, two-server PIR scheme from Fig. 3 in 900 lines of Go and 100 lines of C. We run experiments on a `r7a.metal-48xl` instance on AWS with 192 cores and 1,536 GB of RAM. To measure server time, we report the server throughput (i.e., the number of PIR queries answered in a fixed time interval), measured using sufficiently many threads of execution to saturate the machine. We perform these measurements in two settings: with and without networking (i.e., sending the PIR queries

		This work (Fig. 3)				XOR PIR						
Size n (GB)	**Deg.** D	Up. (bits)	Down. (MB)	Tput, no net. (query/s)	Tput (query/s)	Comm. (MB)	Tput, no net. (query/s)	Tput (query/s)	***Storage blowup***	***Comm. blowup***	***Mem. accesses saved***	***Tput gain***
Databases with 1-byte records.												
122	19	40	356.3	29	6	0.39	12	6	8×	914×	351×	1.0×
82	17	40	95.5	101	18	0.32	24	10	12×	298×	879×	1.8×
37	15	40	22.2	376	121	0.22	42	22	28×	101×	1,707×	5.5×
11	13	40	4.4	1,646	636	0.12	118	71	93×	37×	2,560×	9.0×
2	11	40	0.7	7,620	3,804	0.05	881	374	512×	14×	2,926×	10.2×
0.25	9	40	0.1	43,614	24,252	0.02	7,648	3,673	4,022×	6×	2,560×	6.6×
Databases with 32-byte records.												
135	17	35	984.7	70	2	0.41	9	6	8×	2,402×	140×	0.3×
96	15	35	266.5	287	9	0.35	16	8	11×	761×	369×	1.1×
43	13	35	61.3	1,155	40	0.23	26	18	24×	267×	718×	2.2×
12	11	35	11.7	5,170	234	0.12	119	66	85×	98×	1,050×	3.5×
2	9	35	1.8	19,562	1,541	0.05	1,104	394	512×	36×	1,138×	3.9×
Databases with 64-byte records.												
139	17	34	1,538.8	61	1	0.41	10	5	7×	3,753×	92×	0.2×
110	15	34	430.6	170	5	0.37	11	7	9×	1,164×	262×	0.7×
55	13	34	102.3	853	24	0.26	18	13	19×	393×	551×	1.8×
17	11	34	20.2	3,881	134	0.15	74	47	60×	135×	862×	2.9×
3	9	34	3.2	16,945	867	0.06	627	253	341×	52×	960×	3.4×

Table 2: Performance of our two-server PIR with preprocessing and of XOR PIR, on databases of varying size and record length. Our scheme produces a 1 TB data structure: for each record length ℓ, we set the number of variables m so that the storage $2^m \cdot \ell$ is 1 TB (m is equal to the upload, in column 3), vary the degree D (column 2), and run on the maximal integer database size supported by these parameters (column 1). We report upload and download communication, together with server throughput with and without networking. The storage blowup (column 10) is the ratio of the encoded database size to the original database size. The number of memory accesses saved (column 12) is the ratio between the database size and the download size. The throughput gain (column 13) is the ratio of the throughputs of each scheme, measured with network costs (columns 6 and 9).

and answers over a network between the client and server machine). When including networking, we send queries between two AWS availability regions (`us-east-1` and `us-east-2`). Our code is available at `github.com/ahenzinger/finite-diffs-pir`, and we describe optimizations that it uses and details on our measurements in Appendix E.

Baseline: Prior PIR with preprocessing. In Table 1, we compare the concrete storage overheads in our PIR-with-preprocessing scheme to those of Beimel et al. [BIM00] and Ghoshal et al. [GLM+25]. (Fig. 2 gives the asymptotic version of this comparison.)Relative to prior schemes, on databases that are gigabytes in size, our schemes shrink the server storage by *four to six orders of magnitude*, while incurring the same communication and requiring each server to read the same number of bits to answer each PIR query.

Baseline: Linear-time PIR. Next, we compare the performance of our PIR with preprocessing to the fastest linear-time PIR scheme: *XOR PIR* [CGKS95]. On an n-bit database, XOR PIR requires $4\sqrt{n}$ bits of communication and $O(n)$ server time per query: concretely, the two servers perform one $\mathbb{F}_2$-operation per bit in the database, as we detail in Appendix E. We implement XOR PIR in Go and C, including the "XOR PIR fast" optimization described in [HHCG+23, Section 2].

Storage-time tradeoff. Table 2 reports the cost of our PIR-with-preprocessing scheme on databases of varying size and record length, along with (1) the blowup in server storage, (2) the blowup in communication, (3) the decrease in the number of memory accesses performed to answer each query, and (4) the speedup in server throughput over XOR PIR (including networking). The smaller the ratio of parameters D/m in our scheme, the larger the storage blowup but also the larger the speedups in server throughput.

On an 11 GB database with 1-byte records, encoded into a 1 TB data structure, our servers answer a PIR query by probing and sending back 4.4 MB from this data structure. That is, by blowing up the storage by 93×, our PIR scheme provides a 2,560× decrease in the number of memory accesses per query. Since these memory accesses are RAM accesses to a large array (as opposed to a linear scan) and since the communication is larger than in XOR PIR, this translates to 9× higher server throughput than XOR PIR.

On a 2 GB database with 1-byte records, encoded into a 1 TB data structure, our servers answer a PIR query by probing and sending back 0.7 MB from the data structure. That is, by blowing up the storage by 512×, our PIR scheme provides a 2,926× decrease in the number of memory accesses per query, which translates to 10× higher server throughput than XOR PIR.

When the ratio of parameters D/m is large (approaching $1/2$), the storage overhead in our scheme shrinks, but the communication overhead grows. Because of this large communication cost, our PIR servers become bottlenecked by the rate at which they can send PIR answers over the network. As a result, when the ratio D/m is sufficiently close to $1/2$, our schemes do not show a throughput gain (when including network costs).

Communication. The main limitation of our PIR scheme is its communication. Our PIR is "imbalanced": on database size n, the upload scales as $\Theta(\log n)$ and is concretely tens of bits, while the download scales as $\Theta(n^\epsilon)$ for a constant $1/2 < \epsilon < 1$ and is concretely megabytes to gigabytes(Table 2). In contrast, XOR PIR is "balanced" and has communication $4\sqrt{n}$. Concretely, this is one to three orders of magnitude smaller than the communication of our scheme.

At the same time, as described in Section 3.3, the communication in our scheme could be dramatically reduced using homomorphic encryption, without increasing the number of memory accesses (or the asymptotic server work) per query. As shown in Table 2, in some settings, the server throughput of our scheme without networking is substantially larger than that with networking, indicating that our scheme's end-to-end throughput is bottlenecked by networking costs. As a result, decreasing the communication

in our schemes could improve the server throughput further yet. We leave implementing and optimizing these communication-shrinking techniques for future work. In particular, we expect techniques for speeding up homomorphic encryption in the context of linear-time PIR to be complementary with our protocols [ALP+21b, MW22, HHCG+23, MW24, BMW24, LHCG25].

7 Conclusion

This paper presents information-theoretic PIR with preprocessing with quasilinear server storage and polynomially sublinear server time, using two or more servers. We leave open the question of whether it is possible to construct—or rule out—information-theoretic PIR with preprocessing achieving a better tradeoff between storage and time. Towards this goal, we pose three concrete questions:

1. *Does information-theoretic PIR with preprocessing with a constant number of servers, storage* $n^{1+o(1)}$*, and server time* $n^{1/2-\Omega(1)}$ *exist?* Our results with quasilinear storage achieve server time $n^{1/2+\epsilon}$ (Corollary D.11). In contrast, with $\mathsf{poly}(n)$ storage, protocols with server time n^{ϵ} are known [BIM00, GLM+25].

2. *Does information-theoretic PIR with preprocessing with a constant number of servers, server storage* $\mathsf{poly}(n)$*, server time* $n^{1-\Omega(1)}$*, and communication* $n^{o(1)}$ *exist?* Our schemes only achieve these efficiencies using computational hardness. At the same time, known information-theoretic PIR based on matching vector codes provides subpolynomial communication [DG16], but not with double efficiency. The key challenge with preprocessing these schemes is that they encode the database as a polynomial in $\omega(\log n)$ variables, making it infeasible to store this polynomial's truth table.

3. *Harder yet, does information-theoretic PIR with preprocessing with a constant number of servers, server storage* $\mathsf{poly}(n)$*, and server time* $n^{o(1)}$ *exist?*

A related question is to design better data structures for fast evaluation of a multivariate polynomial f and its derivatives. One promising direction would be to interpolate between the two extremes studied in this work: on the one hand, precomputing only the truth table of the polynomial f and computing its derivatives on-the-fly via finite differences (Section 5.2); on the other hand, precomputing the full truth table of f and of all partial derivatives needed (Section 5.3). For instance, it may be beneficial to precompute only certain evaluations of the polynomial f and certain derivatives, while evaluating others on-the-fly.

Acknowledgements. We are grateful to Vinod Vaikuntanathan, Henry Corrigan-Gibbs, and Yael Tauman Kalai for encouraging this work and providing guidance all along. We thank Nickolai Zeldovich for insightful conversations about our schemes' evaluation and suggesting to measure throughput. We thank Henry Corrigan-Gibbs, Yuval Ishai, Ryan Lehmkuhl, and David Wu for their feedback on early versions of this work. We in particular thank Yuval Ishai for pointing us to the communication-efficient PIR schemes in [BIM04, Theorem 4.7]. We also thank anonymous reviewers for useful comments.

The authors consulted ChatGPT for generating some of the TikZ code that was used in Fig. 2.

References

[AAAG23] Ishtiyaque Ahmad, Divyakant Agrawal, Amr El Abbadi, and Trinabh Gupta. Private information retrieval in large scale public data repositories. *Proceedings of the VLDB Endowment*, 16(12):3868–3871, 2023. 1

[ABG+24] Hilal Asi, Fabian Boemer, Nicholas Genise, Muhammad Haris Mughees, Tabitha Ogilvie, Rehan Rishi, Kunal Talwar, Karl Tarbe, Akshay Wadia, Ruiyu Zhu, et al. Scalable private search with Wally. *arXiv preprint arXiv:2406.06761*, 2024. 1

[ABL24] Bar Alon, Amos Beimel, and Or Lasri. Simplified PIR and CDS protocols and improved linear secret-sharing schemes. *Cryptology ePrint Archive*, 2024. 8

[ACLS18] Sebastian Angel, Hao Chen, Kim Laine, and Srinath T. V. Setty. PIR with compressed queries and amortized query processing. In *S&P*, 2018. 8

[ALP+21a] Asra Ali, Tancrède Lepoint, Sarvar Patel, Mariana Raykova, Phillipp Schoppmann, Karn Seth, and Kevin Yeo. Communication–computation trade-offs in PIR. In *USENIX Security*, pages 1811–1828. USENIX Association, 2021. 8

[ALP+21b] Asra Ali, Tancrède Lepoint, Sarvar Patel, Mariana Raykova, Phillipp Schoppmann, Karn Seth, and Kevin Yeo. Communication–Computation trade-offs in PIR. In *USENIX Security*, 2021. 26

[Amb97] Andris Ambainis. Upper bound on the communication complexity of private information retrieval. In *ICALP*, pages 401–407. Springer, 1997. 8

[AMBFK16] Carlos Aguilar-Melchor, Joris Barrier, Laurent Fousse, and Marc-Olivier Killijian. XPIR: Private information retrieval for everyone. *PoPETs*, 2016. 8

[Art] Apple Developer Article. Getting up-to-date calling and blocking information for your app. https://developer.apple.com/documentation/identitylookup/getting-up-to-date-calling-and-blocking-information-for-your-app. Accessed 6 July 2025. 1

[AS16] Sebastian Angel and Srinath Setty. Unobservable communication over fully untrusted infrastructure. In *OSDI*, 2016. 1, 8

[ASA+21] Ishtiyaque Ahmad, Laboni Sarker, Divyakant Agrawal, Amr El Abbadi, and Trinabh Gupta. Coeus: A system for oblivious document ranking and retrieval. In *SOSP*, 2021. 1

[AYA+21] Ishtiyaque Ahmad, Yuntian Yang, Divyakant Agrawal, Amr El Abbadi, and Trinabh Gupta. Addra: Metadata-private voice communication over fully untrusted infrastructure. In *OSDI*, 2021. 8

[BGG+24] Vishwas Bhargava, Sumanta Ghosh, Zeyu Guo, Mrinal Kumar, and Chris Umans. Fast multivariate multipoint evaluation over all finite fields. *J. ACM*, 71(3):22, 2024. 5, 15, 19, 50

[BGI15] Elette Boyle, Niv Gilboa, and Yuval Ishai. Function secret sharing. In *EUROCRYPT*, pages 337–367. Springer, 2015. 8

[BGI16] Elette Boyle, Niv Gilboa, and Yuval Ishai. Function secret sharing: Improvements and extensions. In *CCS*, pages 1292–1303. ACM, 2016. 1, 8

[BGKM22] Vishwas Bhargava, Sumanta Ghosh, Mrinal Kumar, and Chandra Kanta Mohapatra. Fast, algebraic multivariate multipoint evaluation in small characteristic and applications. In Stefano Leonardi and Anupam Gupta, editors, *STOC '22: 54th Annual ACM SIGACT Symposium on Theory of Computing, Rome, Italy, June 20 - 24, 2022*, pages 403–415. ACM, 2022. 19, 50, 51, 52

[BGN05] Dan Boneh, Eu-Jin Goh, and Kobbi Nissim. Evaluating 2-DNF formulas on ciphertexts. In *TCC*, 2005. 45

[BGV12] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (Leveled) fully homomorphic encryption without bootstrapping. In *ITCS*, 2012. 17

[BI01] Amos Beimel and Yuval Ishai. Information-theoretic private information retrieval: A unified construction. In *ICALP*, pages 912–926. Springer, 2001. 8

[BIK05] Amos Beimel, Yuval Ishai, and Eyal Kushilevitz. General constructions for information-theoretic private information retrieval. *Journal of Computer and System Sciences*, 71(2), 2005. 5, 7, 12, 13

[BIKR02] Amos Beimel, Yuval Ishai, Eyal Kushilevitz, and Jean-François Raymond. Breaking the $O(n^{1/(2k-1)})$ barrier for information-theoretic private information retrieval. In *FOCS*, pages 261–270. IEEE Computer Society, 2002. 5, 7, 8

[BIM00] Amos Beimel, Yuval Ishai, and Tal Malkin. Reducing the servers computation in private information retrieval: PIR with preprocessing. In Mihir Bellare, editor, *Advances in Cryptology - CRYPTO 2000, 20th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 2000, Proceedings*, volume 1880 of *Lecture Notes in Computer Science*, pages 55–73. Springer, 2000. 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 18, 20, 21, 23, 25, 26

[BIM04] Amos Beimel, Yuval Ishai, and Tal Malkin. Reducing the servers' computation in private information retrieval: PIR with preprocessing. *Journal of Cryptology*, 17(2):125–151, February 2004. 17, 26

[BIPW17] Elette Boyle, Yuval Ishai, Rafael Pass, and Mary Wootters. Can we access a database both locally and privately? In *TCC*, pages 662–693. Springer, 2017. 2, 9

[BKMP12] Michael Backes, Aniket Kate, Matteo Maffei, and Kim Pecina. ObliviAd: provably secure and practical online behavioral advertising. In *S&P*, 2012. 1

[BKW19] Andreas Björklund, Petteri Kaski, and Ryan Williams. Generalized Kakeya sets for polynomial evaluation and faster computation of fermionants. *Algorithmica*, 81(10):4010–4028, 2019. 5, 15, 19

[BMW24] Alexander Burton, Samir Jordan Menon, and David J Wu. Respire: High-rate PIR for databases with small records. In *CCS*, 2024. 8, 26

[BS80] Jon Louis Bentley and James B. Saxe. Decomposable searching problems I: static-to-dynamic transformation. *J. Algorithms*, 1980. 11

[BV11] Zvika Brakerski and Vinod Vaikuntanathan. Fully homomorphic encryption from ring-LWE and security for key dependent messages. In *CRYPTO*, 2011. 3, 5, 17, 49

[CG97] Benny Chor and Niv Gilboa. Computationally private information retrieval (extended abstract). In *STOC*, pages 304–313. ACM, 1997. 8

[CGHK22] Henry Corrigan-Gibbs, Alexandra Henzinger, and Dmitry Kogan. Single-server private information retrieval with sublinear amortized time. In *EUROCRYPT*, 2022. 8, 11

[CGHKV25] Henry Corrigan-Gibbs, Alexandra Henzinger, Yael Kalai, and Vinod Vaikuntanathan. Somewhat homomorphic encryption from linear homomorphism and sparse LPN. In *EUROCRYPT*, 2025. 45

[CGK20] Henry Corrigan-Gibbs and Dmitry Kogan. Private information retrieval with sublinear online time. In *EUROCRYPT*, pages 44–75. Springer, 2020. 8, 11

[CGKS95] Benny Chor, Oded Goldreich, Eyal Kushilevitz, and Madhu Sudan. Private information retrieval. In *FOCS*, pages 41–50. IEEE Computer Society, 1995. 1, 3, 4, 8, 9, 10, 20, 25

[Cha04] Yan-Cheng Chang. Single database private information retrieval with logarithmic communication. In *ACISP*, pages 50–61. Springer, 2004. 8

[CHR17] Ran Canetti, Justin Holmgren, and Silas Richelson. Towards doubly efficient private information retrieval. In *TCC*, pages 694–726. Springer, 2017. 2, 9

[CIMR25] Caicai Chen, Yuval Ishai, Tamer Mour, and Alon Rosen. Secret-key PIR from random linear codes. *Cryptology ePrint Archive*, 2025. 1, 9

[CMS99] Christian Cachin, Silvio Micali, and Markus Stadler. Computationally private information retrieval with polylogarithmic communication. In *EUROCRYPT*, pages 402–414. Springer, 1999. 1, 8

[CSM+20] Raymond Cheng, William Scott, Elisaweta Masserova, Irene Zhang, Vipul Goyal, Thomas E. Anderson, Arvind Krishnamurthy, and Bryan Parno. Talek: Private group messaging with hidden access patterns. In *ACSAC*, pages 84–99. ACM, 2020. 1

[DCG23] Emma Dauterman and Henry Corrigan-Gibbs. Lightweb: Private web browsing without all the baggage. In *HotNets*, 2023. 1

[DDLO25] Nico Döttling, Jesko Dujmovic, Julian Loss, and Maciej Obremski. Minicrypt PIR for big batches. *Cryptology ePrint Archive*, 2025. 8

[DFL+20] Emma Dauterman, Eric Feng, Ellen Luo, Raluca Ada Popa, and Ion Stoica. DORY: an encrypted search system with distributed trust. In *OSDI*, pages 1101–1119. USENIX Association, 2020. 1

[DG16] Zeev Dvir and Sivakanth Gopi. 2-server PIR with subpolynomial communication. *J. ACM*, 2016. 1, 8, 12, 18, 26

[DGI+19] Nico Döttling, Sanjam Garg, Yuval Ishai, Giulio Malavolta, Tamer Mour, and Rafail Ostrovsky. Trapdoor hash functions and their applications. In *CRYPTO*, pages 3–32. Springer, 2019. 1, 3, 8

[DJ01] Ivan Damgård and Mads Jurik. A generalisation, a simplification and some applications of Paillier's probabilistic public-key system. In *PKC*, 2001. 17

[DPC23] Alex Davidson, Gonçalo Pestana, and Sofía Celi. FrodoPIR: Simple, scalable, single-server private information retrieval. *PETS*, 2023. 8

[Efr12] Klim Efremenko. 3-query locally decodable codes of subexponential length. *SIAM J. Comput.*, 2012. 8

[ElG85] Taher ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Trans. Inf. Theory*, 31(4):469–472, 1985. 3, 17

[FKW25] Nick Fischer, Melvin Kallmayer, and Leo Wennmann. A simple algorithm for trimmed multipoint evaluation. *arXiv preprint arXiv:2507.00196*, 2025. 42

[For14] Michael A. Forbes. *Polynomial identity testing of read-once oblivious algebraic branching programs*. PhD thesis, Massachusetts Institute of Technology, Cambridge, MA, USA, 2014. 50

[FOSZ23] Brett Hemenway Falk, Rafail Ostrovsky, Matan Shtepel, and Jacob Zhang. GigaDORAM: Breaking the billion address barrier. 2023. 9

[FV12] Junfeng Fan and Frederik Vercauteren. Somewhat practical fully homomorphic encryption. *Cryptology ePrint Archive*, 2012. 17

[GCM+16] Trinabh Gupta, Natacha Crooks, Whitney Mulhern, Srinath Setty, Lorenzo Alvisi, and Michael Walfish. Scalable and private media consumption with Popcorn. In *NSDI*, pages 91–107, 2016. 1

[Gen09] Craig Gentry. A fully homomorphic encryption scheme. Ph.D. thesis, Stanford University, 2009. 3, 17, 49

[GH19] Craig Gentry and Shai Halevi. Compressible FHE with applications to PIR. In *TCC*, 2019. 8

[GHV10] Craig Gentry, Shai Halevi, and Vinod Vaikuntanathan. A simple BGN-type cryptosystem from LWE. In *EUROCRYPT*, 2010. 45

[GI14] Niv Gilboa and Yuval Ishai. Distributed point functions and their applications. In *EUROCRYPT*, pages 640–658. Springer, 2014. 8

[GKS25] Fatemeh Ghasemi, Swastik Kopparty, and Madhu Sudan. Improved PIR schemes using matching vectors and derivatives. In *STOC*, 2025. 1, 8

[GLM16] Matthew Green, Watson Ladd, and Ian Miers. A protocol for privately reporting ad impressions at scale. In *CCS*, pages 1591–1601. ACM, 2016. 1

[GLM+25] Ashrujit Ghoshal, Baitian Li, Yaohua Ma, Chenxin Dai, and Elaine Shi. Scalable multi-server private information retrieval. In *TCC*. Springer, 2025. 1, 2, 3, 4, 5, 6, 7, 8, 9, 12, 13, 15, 19, 20, 21, 23, 25, 26, 55

[GM84] Shafi Goldwasser and Silvio Micali. Probabilistic encryption. *Journal of computer and system sciences*, 1984. 3, 17

[GR05] Craig Gentry and Zulfikar Ramzan. Single-database private information retrieval with constant communication rate. In *ICALP*, pages 803–815. Springer, 2005. 1, 8

[GSW13] Craig Gentry, Amit Sahai, and Brent Waters. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In *CRYPTO*, 2013. 17

[GZS24] Ashrujit Ghoshal, Mingxun Zhou, and Elaine Shi. Efficient pre-processing PIR without public-key cryptography. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 210–240. Springer, 2024. 8

[Hal17] Shai Halevi. Homomorphic encryption. In Yehuda Lindell, editor, *Tutorials on the Foundations of Cryptography*, pages 219–276. Springer International Publishing, 2017. 17, 45

[HDCGZ23] Alexandra Henzinger, Emma Dauterman, Henry Corrigan-Gibbs, and Nickolai Zeldovich. Private web search with Tiptoe. In *SOSP*, 2023. 1, 8

[Hen16] Ryan Henry. Polynomial batch codes for efficient IT-PIR. *PoPETs*, 2016. 8

[HHCG+23] Alexandra Henzinger, Matthew M. Hong, Henry Corrigan-Gibbs, Sarah Meiklejohn, and Vinod Vaikuntanathan. One server for the price of two: Simple and fast single-server private information retrieval. In *USENIX Security*, 2023. 1, 8, 25, 26

[HOG11] Ryan Henry, Femi G. Olumofin, and Ian Goldberg. Practical PIR for electronic commerce. In *CCS*, pages 677–690. ACM, 2011. 1

[HOWW19] Ariel Hamlin, Rafail Ostrovsky, Mor Weiss, and Daniel Wichs. Private anonymous data access. In *EUROCRYPT*, 2019. 11

[HPPY25] Alexander Hoover, Sarvar Patel, Giuseppe Persiano, and Kevin Yeo. Plinko: single-server PIR with efficient updates via invertible PRFs. In *EUROCRYPT*, 2025. 8

[HPW25] Lena Heimberger, Christopher Patton, and Bas Westerbaan. Private sct auditing, revisited. *Cryptology ePrint Archive*, 2025. 1

[HSW23] Laura Hetz, Thomas Schneider, and Christian Weinert. Scaling mobile private contact discovery to billions of users. In *European Symposium on Research in Computer Security*, pages 455–476. Springer, 2023. 1

[IKLM24] Yuval Ishai, Mahimna Kelkar, Daniel Lee, and Yiping Ma. Information-theoretic single-server PIR in the shuffle model. In *ITCS*, 2024. 8

[IKOS04] Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, and Amit Sahai. Batch codes and their applications. In *STOC*, pages 262–271. ACM, 2004. 8, 45, 46

[IKOS06] Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, and Amit Sahai. Cryptography from anonymity. In *FOCS*, 2006. 8

[ILM21] Yuval Ishai, Russell WF Lai, and Giulio Malavolta. A geometric approach to homomorphic secret sharing. In *Public-Key Cryptography*, 2021. 4, 15

[ISW24] Yuval Ishai, Elaine Shi, and Daniel Wichs. PIR with client-side preprocessing: information-theoretic constructions and lower bounds. In *Annual International Cryptology Conference*, pages 148–182. Springer, 2024. 8

[Jue01] Ari Juels. Targeted advertising ... and privacy too. In *CT-RSA*, pages 408–424. Springer, 2001. 1

[KCG21] Dmitry Kogan and Henry Corrigan-Gibbs. Private blocklist lookups with Checklist. In *USENIX Security*, 2021. 1, 8, 11

[KO97] Eyal Kushilevitz and Rafail Ostrovsky. Replication is not needed: Single database, computationally-private information retrieval. In *FOCS*, pages 364–373. IEEE, 1997. 1, 8, 17, 45, 46

[Kop15] Swastik Kopparty. Some remarks on multiplicity codes. *arXiv preprint arXiv:1505.07547*, 2015. 5

[KRAP25] Darya Kaviani, Deevashwer Rathee, Bhargav Annem, and Raluca Ada Popa. Myco: Unlocking polylogarithmic accesses in metadata-private messaging. In *2025 IEEE Symposium on Security and Privacy (SP)*, pages 4419–4437. IEEE, 2025. 1

[KT00] Jonathan Katz and Luca Trevisan. On the efficiency of local decoding procedures for error-correcting codes. In *STOC*, 2000. 18

[KU11] Kiran S. Kedlaya and Christopher Umans. Fast polynomial factorization and modular composition. *SIAM J. Comput.*, 40(6):1767–1802, 2011. 52

[Lee25] Keewoo Lee. Barely doubly-efficient SimplePIR. Cryptology ePrint Archive, Paper 2025/1305, 2025. 1

[LG15] Wouter Lueks and Ian Goldberg. Sublinear scaling for multi-client private information retrieval. In *Financial Cryptography*, pages 168–186. Springer, 2015. 8

[LHCG25] Ryan Lehmkuhl, Alexandra Henzinger, and Henry Corrigan-Gibbs. Distributional private information retrieval. In *USENIX Security*, 2025. 1, 26

[Lip05] Helger Lipmaa. An oblivious transfer protocol with log-squared communication. In *International Conference on Information Security*, pages 314–328. Springer, 2005. 1, 8

[Lip09] Helger Lipmaa. First CPIR protocol with data-dependent computation. In *ICISC*, 2009. 1

[LJX+24] Maximilian Lam, Jeff Johnson, Wenjie Xiong, Kiwan Maeng, Udit Gupta, Yang Li, Liangzhen Lai, Ilias Leontiadis, Minsoo Rhu, Hsien-Hsin S. Lee, Vijay Janapa Reddi, Gu-Yeon Wei, David Brooks, and Edward Suh. GPU-based private information retrieval for on-device machine learning inference. page 197–214, 2024. 1

[LLF+25] Arthur Lazzaretti, Zeyu Liu, Ben Fisch, Peihan Miao, and Charalampos Papamanthou. Multi-server doubly efficient PIR in the classical model and beyond. In *TCC*. Springer, 2025. 1, 7, 9

[LLMT25] Chengyu Lin, Zeyu Liu, Peihan Miao, and Max Tromanhauser. Finding balance in unbalanced PSI: A new construction from single-server PIR. *IACR Communications in Cryptology*, 2(1), 2025. 1

[LMRSW24] Baiyu Li, Daniele Micciancio, Mariana Raykova, and Mark Schultz-Wu. Hintless single-server private information retrieval. In *CRYPTO*, 2024. 8

[LMW23] Wei-Kai Lin, Ethan Mook, and Daniel Wichs. Doubly efficient private information retrieval and fully homomorphic RAM computation from ring LWE. In *STOC*, 2023. 1, 2, 5, 9, 11

[LMW25] Wei-Kai Lin, Ethan Mook, and Daniel Wichs. Black box crypto is useless for doubly efficient PIR. In *EUROCRYPT*, 2025. 1

[LP23] Arthur Lazzaretti and Charalampos Papamanthou. Near-optimal private information retrieval with preprocessing. In *Theory of Cryptography Conference*, pages 406–435. Springer, 2023. 8

[MCR21] Muhammad Haris Mughees, Hao Chen, and Ling Ren. OnionPIR: Response efficient single-server PIR. In *CCS*, 2021. 8

[MOT+11] Prateek Mittal, Femi Olumofin, Carmela Troncoso, Nikita Borisov, and Ian Goldberg. PIR-Tor: Scalable anonymous communication using private information retrieval. In *USENIX Security*, 2011. 1

[MVM25] Miti Mazmudar, Shannon Veitch, and Rasoul Akhavan Mahdavi. Peer2PIR: Private queries for IPFS. In *2025 IEEE Symposium on Security and Privacy (SP)*, 2025. 1

[MW22] Samir Jordan Menon and David J. Wu. Spiral: Fast, high-rate single-server PIR via FHE composition. In *S&P*, 2022. 1, 8, 26

[MW24] Samir Jordan Menon and David J. Wu. YPIR: High-Throughput Single-Server PIR with silent preprocessing. In *USENIX Security*, 2024. 8, 26

[OPPW24] Hiroki Okada, Rachel Player, Simon Pohmann, and Christian Weinert. Towards practical doubly-efficient private information retrieval. In *International Conference on Financial Cryptography and Data Security*, 2024. 2, 9

[OS97] Rafail Ostrovsky and Victor Shoup. Private information storage. In *STOC*, 1997. 11

[Pai99] Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *EUROCRYPT*, 1999. 3, 17

[PPY18] Sarvar Patel, Giuseppe Persiano, and Kevin Yeo. Private stateful information retrieval. In *Proceedings of the 2018 ACM SIGSAC conference on computer and communications security*, pages 1002–1019, 2018. 8

[PY22] Giuseppe Persiano and Kevin Yeo. Limits of preprocessing for single-server PIR. In *SODA*, 2022. 9

[RAD78] Ronald L Rivest, Len Adleman, and Michael L Dertouzos. On data banks and privacy homomorphisms. *Foundations of secure computation*, 4(11), 1978. 3, 17, 49

[Reg09] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. *J. ACM*, 2009. 3

[RMS24] Ling Ren, Muhammad Haris Mughees, and I Sun. Simple and practical amortized sublinear private information retrieval using dummy subsets. In *Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security*, pages 1420–1433, 2024. 8

[SACM21] Elaine Shi, Waqar Aqeel, Balakrishnan Chandrasekaran, and Bruce Maggs. Puncturable pseudorandom sets and private information retrieval with near-optimal online bandwidth and time. In *CRYPTO*. Springer-Verlag, 2021. 8

[Sha48] Claude E. Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27(3), 1948. 35

[SSHD21] Sacha Servan-Schreiber, Kyle Hogan, and Srinivas Devadas. AdVeil: A private targeted-advertising ecosystem. Cryptology ePrint Archive, Report 2021/1032, 2021. 1

[TWG24] Elkana Tovey, Jonathan Weiss, and Yossi Gilad. Distributed PIR: Scaling private messaging via the users' machines. In *Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security*, pages 1967–1981, 2024. 1

[vdHS12] Joris van der Hoeven and Éric Schost. Multi-point evaluation in higher dimensions. *Applicable Algebra in Engineering, Communication and Computing*, 24(1):37–52, December 2012. 42

[vzGG03] Joachim von zur Gathen and Jürgen Gerhard. *Modern computer algebra (2. ed.)*. Cambridge University Press, 2003. 5, 19, 51

[WR25] Zhikun Wang and Ling Ren. Single-server client preprocessing PIR with tight space-time trade-off. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 94–122. Springer, 2025. 8

[WY05] David Woodruff and Sergey Yekhanin. A geometric approach to information-theoretic private information retrieval. In *CCC*. IEEE, 2005. 1, 4, 5, 12

[Yek08] Sergey Yekhanin. Towards 3-query locally decodable codes of subexponential length. *J. ACM*, 2008. 8

[Yek12] Sergey Yekhanin. Locally decodable codes. *Foundations and Trends in Theoretical Computer Science*, 6(3), 2012. 5, 18

[Yeo23] Kevin Yeo. Private information retrieval: Improvements and limitations. https://www.youtube.com/watch?v=8eqJztvaT1w, 2023. Accessed 26 June 2025. 1

[ZLTS23] Mingxun Zhou, Wei-Kai Lin, Yiannis Tselekounis, and Elaine Shi. Optimal single-server private information retrieval. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 395–425. Springer, 2023. 8

[ZMA22] Ke Zhong, Yiping Ma, and Sebastian Angel. Ibex: Privacy-preserving ad conversion tracking and bidding. In *ACM SIGSAC Conference on Computer and Communications Security*, 2022. 1

[ZPZS24] Mingxun Zhou, Andrew Park, Wenting Zheng, and Elaine Shi. Piano: extremely simple, single-server PIR with sublinear server computation. In *2024 IEEE Symposium on Security and Privacy (SP)*, 2024. 8

[Zve13] Jan Zverina. U.s. media consumption to rise to 15.5 hours a day – per person – by 2015. https://today.ucsd.edu/story/u.s.mediaconsumptiontoriseto15.5hoursadayperpersonby2015, 2013. Accessed 23 June 2025. 1

A Asymptotics of Count

The goal of this section is to prove the following theorem:

Definition A.1. For a constant $d \in \mathbb{N}$ and real number $\mu \in (0, d)$, define $q_\mu(x) = \sum_{i=0}^{d} (\mu - i) \cdot x^i$.

Theorem A.2 (Asymptotics of Count)**.** *Let d be a constant positive integer, let θ be a real in $(0, d)$, and assume the integer m grows. Then, $q_\theta(\cdot)$ has exactly one positive real root, which we denote by $p := p(\theta)$. Moreover, for any $D = (\theta \pm o(1)) \cdot m$, the following hold:*

1. *We have* $\log \mathsf{Count}(D, d, m) = (\mathsf{CntExp}(\theta, d) + o(1)) \cdot m$*, where we define*

$$\mathsf{CntExp}(\theta, d) := \log\left(1 + p + \ldots + p^d\right) - \theta \log p.$$

2. *If $\theta \le d/2$, then* $\log \mathsf{Count}_{\le}(D, d, m) = (\mathsf{CntExp}(\theta, d) + o(1)) \cdot m$*. Otherwise* $\log \mathsf{Count}_{\le}(D, d, m) = (\log(d + 1) + o(1)) \cdot m$*.*

   For convenience, we will let $\mathsf{CntExp}_{\le}(\theta, d)$ *denote* $\mathsf{CntExp}(\theta, d)$ *if* $\theta \le d/2$ *and* $\log(d + 1)$ *otherwise.*

Throughout, we use the following, extended definition of binary entropy:

Definition A.3 (Binary Entropy [Sha48])**.** For any vector $(p_1, \ldots, p_k) \in [0, 1]^k$ such that $\sum_{i=1}^{k} p_i = 1$, the binary entropy H is defined as

$$H(p_1, \ldots, p_k) = -\sum_{i=1}^{k} p_i \log p_i.$$

For a real number $p \in [0, 1]$, we overload notation and additionally define $H(p) = H(p, 1 - p)$. We adopt the convention that $-p \log p$ denotes $\lim_{p \to 0} -p \log p = 0$ when $p = 0$.

Finally, for vectors $\mathbf{a} = (a_1, \ldots, a_m)$ and $\mathbf{e} = (e_1, \ldots, e_m) \in \mathbb{N}^m$, we denote

$$\binom{\mathbf{a}}{\mathbf{e}} = \prod_{i=1}^{m} \binom{a_i}{e_i}.$$

Relating Multinomial Coefficients to Entropy.

Lemma A.4. *Let $\alpha_1, \ldots, \alpha_r \in [0, 1]$ be such that $\sum_i \alpha_i = 1$. Assume $r, \{\alpha_i\}$ are constant and m grows. Then for all k_i such that $k_1 + \ldots + k_r = m$ and $k_i = (\alpha_i + o(1/\log m))m$, we have:*

$$\log \binom{m}{k_1, \ldots, k_r} = (H(\alpha_1, \ldots, \alpha_r) + o(1))m.$$

Proof. Let $g(\alpha)$ be $1/\alpha$ if $\alpha > 0$ and 0 if $\alpha = 0$. We will make use of Stirling's approximation, letting ln denote the base-e logarithm. In the below, we will use $O(T)$ to denote an arbitrary real number (that can be thought of as an error term) with magnitude bounded by some constant times T.

$$\begin{aligned}
\ln \binom{m}{k_1, \ldots, k_r} &= \ln m! - \sum_{i=1}^{r} \ln(k_i!) \\
&= (m \ln m - m + O(\ln m)) - \sum_{i=1}^{r} (k_i \ln k_i - k_i + O(\ln k_i))
\end{aligned}$$

$$
\begin{aligned}
&= m \ln m - \sum_{i=1}^{r} k_i \ln k_i + O(\ln m) \\
&= m \ln m - \sum_{i=1}^{r} (\alpha_i + o(1/\ln m)) m \ln k_i + O(\ln m) \\
&= m \ln m - \sum_{i=1}^{r} \alpha_i m \ln k_i + o(m) \\
&= m \ln m - \sum_{i=1}^{r} \alpha_i m \left(\ln m + \ln(\alpha_i + o(1/\ln m)\right) + o(m) \\
&= -\sum_{i=1}^{r} \alpha_i m \ln(\alpha_i + o(1/\ln m)) + o(m) \\
&= -\sum_{i=1}^{r} \alpha_i m \left[\ln \alpha_i + o(g(\alpha_i)/\ln m)\right] + o(m) \\
&= \ln 2 \cdot H(\alpha_1, \ldots, \alpha_r) m + \frac{m}{\ln m} \cdot \left(\sum_{i=1}^{r} \alpha_i g(\alpha_i) \right) + o(m) \\
&= (\ln 2 \cdot H(\alpha_1, \ldots, \alpha_r) + o(1)) m,
\end{aligned}
$$

and now dividing both sides by $\ln 2$ yields the desired result. □

Connecting Count **to Maximum Entropy.**

Definition A.5. For constant d and $\mu \in (0, d)$, we will define

$$
\beta(\mu) := \max_{\substack{\alpha_0, \ldots, \alpha_d \\ \sum_i \alpha_i = 1 \\ \sum_i i\alpha_i = \mu}} H(\alpha_0, \ldots, \alpha_d).
$$

Equivalently, $\beta(\mu)$ is the maximum entropy of any random variable X supported on $\{0, 1, \ldots, d\}$ with mean μ.

For every vector $\mathbf{k} = (k_0, \ldots, k_d)$ of non-negative integers such that $\sum_i k_i = m$ and $\sum_i i k_i = D$, let $T_{D,d,m,\mathbf{k}}$ be the collection of vectors $\mathbf{x} = (x_1, \ldots, x_m) \in \{0, 1, \ldots, d\}^m$ such that $x_1 + \ldots + x_m = D$, and for each $i \in [0, d]$ the number of times i appears in $\mathbf{x}$ is equal to k_i. Note that

$$
\left| T_{D,d,m,\mathbf{k}} \right| = \binom{m}{k_0, \ldots, k_d}.
$$

We first have:

Lemma A.6. *Define* $M = \max_{\mathbf{k}} \binom{m}{k_0, \ldots, k_d}$, *then we have*

$$
\log \mathsf{Count}(D, d, m) \in [\log M, \log M + o(m)].
$$

Proof. It is evident that we have

$$
\mathsf{Count}(D, d, m) = \sum_{\mathbf{k}} \left| T_{D,d,m,\mathbf{k}} \right| = \sum_{\mathbf{k}} \binom{m}{k_0, \ldots, k_d}.
$$

Moreover, the number of such vectors $\mathbf{k}$ is clearly at most $(m+1)^{d+1} = \mathsf{poly}(m) = \exp(o(m))$. Thus we have:

$$\begin{aligned} &\mathsf{Count}(D, d, m) \in [M, \exp(o(m)) \cdot M] \\ \Rightarrow &\log \mathsf{Count}(D, d, m) \in [\log M, \log M + o(m)], \end{aligned}$$

as desired. □

Lemma A.7. *Recall as in Definition A.5 that*

$$\beta(D/m) = \max_{\substack{\alpha_0,\dots,\alpha_d \\ \sum_i \alpha_i = 1 \\ \sum_i i\alpha_i = D/m}} H(\alpha_0, \dots, \alpha_d).$$

Assume further that there exist $\alpha_0, \dots, \alpha_d$ maximizing this entropy such that $\alpha_0 > 0$. Then we have

$$\log M = (\beta(D/m) + o(1))m.$$

Proof. We will use β to denote $\beta(D/m)$. First, we show that $\log M \le (\beta + o(1))m$. Indeed, let $\mathbf{k}$ be such that $\binom{m}{k_0,\dots,k_d}$ is maximized. Then, setting $\alpha_i = k_i/m$ for all i, we have:

$$\begin{aligned} \log M &= \log \binom{m}{k_0, \dots, k_d} \\ &\le (H(\alpha_0, \dots, \alpha_d) + o(1))\, m \\ &\le (\beta + o(1))m. \end{aligned}$$

Conversely, let $\alpha_0, \dots, \alpha_d$ be such that $\sum_i \alpha_i = 1, \sum_i i\alpha_i = D/m$, and $H(\alpha_0, \dots, \alpha_d) = \beta$. Additionally, we assume this maximizing point is such that $\alpha_0 > 0$. Now define the k_i's as follows:

$$k_i = \begin{cases} \lfloor \alpha_i m \rfloor, & \text{if } i \ge 2, \\ D - \sum_{j=2}^{d} j k_j, & \text{if } i = 1, \text{ and} \\ m - \sum_{j=1}^{d} k_j, & \text{if } i = 0. \end{cases}$$

It is clear that $\sum_i k_i = m$ and $\sum_i i k_i = D$. Secondly, we know $k_i \ge 0$ and $k_i \in \alpha_i m + O(1)$ for all $i \ge 2$. It remains to check the same for $i = 1, 0$. For $i = 1$, we have:

$$\begin{aligned} k_1 = D - \sum_{j=2}^{d} j k_j \ge D - \sum_{j=2}^{d} j\alpha_j m = \alpha_1 m, \text{ and} \\ k_1 \le D - \sum_{j=2}^{d} j(\alpha_j m - 1) = \alpha_1 m + O(1). \end{aligned}$$

For $i = 0$, we have:

$$k_0 = m - \sum_{j=1}^{d} k_j = m - \sum_{j=1}^{d} \big(\alpha_j m + O(1)\big) \in \alpha_0 m + O(1).$$

Since we are assuming that $\alpha_0 > 0$, it follows that $k_0 \ge 0$. To finish, note that:

$$\log M \ge \log \binom{m}{k_0, \dots, k_d}$$

$$= (H(\alpha_0, \ldots, \alpha_d) + o(1))\, m \text{ (Lemma A.4)}$$
$$= (\beta + o(1))m,$$

which concludes our proof. □

Maximizing Entropy. We begin with a little more setup: for any $\mu \in (0, d)$, define the univariate polynomial $q_\mu(x)$ by

$$q_\mu(x) = \sum_{i=0}^{d} (\mu - i) x^i.$$

Lemma A.8 (Existence and Uniqueness of Root)**.** *For any $\mu \in (0, d)$, $q_\mu(x)$ has a unique positive real root.*

Proof. Note that:

$$q_\mu(x) = 0 \Leftrightarrow \mu = \frac{\sum_{i=0}^{d} i x^i}{\sum_{i=0}^{d} x^i}.$$

We claim that the RHS is strictly increasing as a function of $x > 0$. Indeed, its derivative is proportional to:

$$\left(\sum_{i=0}^{d} x^i\right) \cdot \left(\sum_{j=0}^{d-1} (j+1)^2 x^j\right) - \left(\sum_{i=0}^{d} i x^i\right) \cdot \left(\sum_{j=0}^{d-1} (j+1) x^j\right)$$
$$= \sum_{k=0}^{2d-1} (A_k - B_k)\, x^k,$$

where

$$A_k = \sum_{j=\max(k-d,0)}^{\min(k,d-1)} (j+1)^2, \text{ and}$$
$$B_k = \sum_{j=\max(k-d,0)}^{\min(k,d-1)} (k-j)(j+1).$$

We will argue that $A_k \geq B_k$ for all k, and proceed in two cases:

- When $k \leq d-1$, we have $A_k = \sum_{j=0}^{k} (j+1)^2 = 1^2 + \ldots + (k+1)^2$, and

$$B_k = \sum_{j=0}^{k} (k-j)(j+1)$$
$$\leq k \cdot 1 + (k-1) \cdot 2 + \ldots + 1 \cdot k$$
$$\leq 1^2 + 2^2 + \ldots + k^2 \text{ (rearrangement inequality)}$$
$$< A_k.$$

- When $k \geq d$, we have $A_k = \sum_{j=k-d}^{d-1} (j+1)^2 = (k-d+1)^2 + \ldots + d^2$, and

$$B_k = \sum_{j=k-d}^{d-1} (k-j)(j+1)$$
$$= d \cdot (k-d+1) + (d-1) \cdot (k-d+2) + \ldots + (k-d+1) \cdot d$$
$$\leq A_k \text{ (rearrangement inequality).}$$

Since $A_k \geq B_k$ for all k and $A_k > B_k$ for at least one k, it follows that $\frac{\sum_{i=0}^{d} i x^i}{\sum_{i=0}^{d} x^i}$ is strictly increasing as a function of $x > 0$. At $x = 0$, this function evaluates to 0 and as $x \to \infty$, this function approaches d. Thus the conclusion follows by the intermediate value theorem. □

From now on, we let $p(\mu)$ be the unique positive real root x of $q_\mu(x) = 0$. We note that $\lim_{\mu \to 0} p(\mu) = 0, \lim_{\mu \to d} p(\mu) = \infty$, and it is straightforward to verify that $p(d/2) = 1$.

Lemma A.9 (Maximizing Entropy)**.** *For any $\mu \in (0, d)$, let $p := p(\mu)$. Then,*

$$\beta(\mu) = \max_{\substack{\alpha_0, \ldots, \alpha_d \\ \sum_i \alpha_i = 1 \\ \sum_i i\alpha_i = \mu}} H(\alpha_0, \ldots, \alpha_d) = \log\left(1 + p + \ldots + p^d\right) - \mu \log p.$$

Moreover, this maximum is attained at a vector such that $\alpha_0, \alpha_1, \ldots, \alpha_d > 0$.

Proof. For each $i \in [0, d]$, define $\gamma_i = a p^i$, where $a = \frac{1}{1 + p + \ldots + p^d}$. Then:

$$\begin{aligned}
\sum_{i=0}^{d} \gamma_i &= 1, \text{ and} \\
\sum_{i=0}^{d} i\gamma_i &= a \sum_{i=0}^{d} i p^i \\
&= a\mu \sum_{i=0}^{d} p^i \text{ (since } p \text{ is a root of } q_\mu) \\
&= \mu.
\end{aligned}$$

By Gibbs' inequality, we have that:

$$\begin{aligned}
H(\alpha_0, \ldots, \alpha_d) &\leq -\sum_{i=0}^{d} \alpha_i \log \gamma_i \\
&= -\sum_{i=0}^{d} \alpha_i (\log a + i \log p) \\
&= -\log a - \mu \log p.
\end{aligned}$$

So, $\beta(\mu) \leq -\log a - \mu \log p$. Equality follows from noticing that equality holds when $\alpha_i = \gamma_i \forall i$. Note that $\gamma_i > 0$ for all i, so this establishes the final claim. □

Properties of Maximum Entropy: Monotonicity and Stability. Throughout this section, we will view the characterization of $\beta(\cdot)$ in terms of random variables specified in Definition A.5.

Lemma A.10 (Symmetry and Monotonicity)**.** *For all $\mu \in (0, d)$, we have $\beta(\mu) = \beta(d - \mu)$. Moreover, the function $\beta(\cdot)$ is weakly increasing on $(0, d/2)$.*

Proof. We first verify the symmetry property. Consider the mapping that takes a random variable X and maps it to $d - X$. This preserves entropy and the support on $\{0, 1, \ldots, d\}$. It maps $\mathbb{E}[X] \mapsto d - \mathbb{E}[X]$, so it follows that $\beta(\mu) = \beta(d - \mu)$.

Next we address monotonicity. Consider any $0 < \mu < \mu' < d/2$. Let X be a random variable supported on $\{0, 1, \dots, d\}$ such that $\mathbb{E}[X] = \mu$. It suffices to find a random variable X' such that $\mathbb{E}[X'] = \mu'$ and $H(X') \geq H(X)$.

To this end, let Y be a uniform random variable on $\{0, 1, \dots, d\}$, and let $\delta \in (0, 1)$ be such that $\mu' = \delta\mu + (1 - \delta)d/2$. We will take X' to be a convex combination of X and Y i.e. it samples from X with probability δ and from Y with probability $1 - \delta$. It is clear that $\mathbb{E}[X'] = \mu'$. Moreover, since entropy is concave, we have:

$$H(X') \geq \delta H(X) + (1 - \delta)H(Y) \geq H(X),$$

where the second step follows from the fact that the uniform distribution (i.e. Y) maximizes entropy for a given support set. □

Lemma A.11 (Stability)**.** *Let $\mu \in (0, d)$ be some constant. Then, for any $\mu' \in \mu \pm o(1)$, we have:*

$$\left|\beta(\mu) - \beta(\mu')\right| \leq o_\mu(1).$$

Proof. We clearly need only address the case where $\mu \neq \mu'$. Let K be a compact and convex set of probability distributions supported on $\{0, 1, \dots, d\}$ containing X, subject to the following constraints:

- K includes all random variables Y supported on $\{0, 1, \dots, d\}$ such that $|\mathbb{E}[Y] - \mu| \leq \epsilon$ and $\Pr[Y = i] \geq \delta$ for all $i \in [0, d]$, where we take $\epsilon, \delta > 0$ to be sufficiently small constants that could depend on μ.
- For every random variable Y in K, for all $i \in [0, d]$, $\Pr[Y = i] \geq \delta$.
- For every $\nu \in \mu \pm \epsilon$, there exists a random variable Y such that $\mathbb{E}[Y] = \nu$ and such that $H(Y) = \beta(\nu)$.

It follows from Lemma A.9 that such K exists. Note that K only depends on μ. Moreover, because of the second constraint, the binary entropy function $H(\cdot)$ is differentiable everywhere on K and by compactness it must furthermore be $O_\mu(1)$-Lipschitz.

Let X, X' be random variables on $\{0, 1, \dots, d\}$ with maximum entropy subject to $\mathbb{E}[X] = \mu$ and $\mathbb{E}[X'] = \mu'$. Note that $X, X' \in K$. We now argue that $\beta(\mu') \geq \beta(\mu) - o_\mu(1)$; an analogous argument will establish the reverse inequality and thus complete our proof. To this end, let $Y \in K$ be arbitrary so that μ' is strictly contained in the interval between $\mathbb{E}[Y]$ and μ and $|\mathbb{E}[Y] - \mu| = \Omega(1)$, and let $\gamma \in (0, 1)$ be such that $\mu' = (1 - \gamma)\mu + \gamma\mathbb{E}[Y]$. Now let Z be a convex combination of X and Y that samples from X with probability $1 - \gamma$ and Y with probability γ; note that $H(Z) = \mu'$. Then we have:

$$\begin{aligned}
|\gamma| &= \frac{|\mu' - \mu|}{|\mathbb{E}[Y] - \mu|} \\
&\leq O(|\mu' - \mu|) \\
&= o(1), \text{ and} \\
|H(Z) - H(X)| &\leq O_\mu(1) \cdot \|Z - X\| \text{ (Lipschitzness of } H(\cdot) \text{ on } K) \\
&= O_\mu(\gamma) \cdot \|Y - X\| \\
&\leq o_\mu(1) \\
\Rightarrow H(Z) &\geq H(X) - o_\mu(1) \\
&= \beta(\mu) - o_\mu(1) \\
\Rightarrow \beta(\mu') &\geq \beta(\mu) - o_\mu(1).
\end{aligned}$$

An analogous argument applied to X' instead of X yields the conclusion. □

Finally, we can put everything together to prove Theorem A.2:

Proof of Theorem A.2. First we address the growth of $\log \mathsf{Count}(D, d, m)$. By combining Lemmas A.6, A.7 and A.11, we have

$$\log \mathsf{Count}(D, d, m) = (\beta(D/m) + o(1))\, m = (\beta(\theta) + o(1))\, m.$$

Secondly, note that:

$$\begin{aligned}
\mathsf{Count}_{\leq}(D, d, m) &= \sum_{D' \leq D} \mathsf{Count}(D', d, m) \in [1, D] \cdot \max_{D' \leq D} \mathsf{Count}(D', d, m) \\
\Rightarrow \log \mathsf{Count}_{\leq}(D, d, m) &= \max_{D' \leq D} \log \mathsf{Count}(D', d, m) + O(\log m) \\
&= \max_{D' \leq D} (\beta(D'/m) + o(1))m \text{ (Lemmas A.6 and A.7)} \\
&= (\beta(\min(D/m, d/2)) + o(1))m \text{ (Lemma A.10)} \\
&= (\beta(\min(\theta, d/2)) + o(1))m.
\end{aligned}$$

Finally, since $p(d/2) = 1$, it follows from Lemma A.9 that:

$$\beta(d/2) = \log(d+1) - \mu \log 1 = \log(d+1),$$

which completes the proof of the theorem. □

Remark A.12 (Special case of $d = 1$ in Theorem A.2). When $d = 1$, we have $q_\theta(x) = \theta + (\theta - 1)x \Rightarrow p(\theta) = \frac{\theta}{1-\theta}$. Then,

$$\mathsf{CntExp}(\theta, d) = \log(1 + p) - \theta \log p = \log \frac{1}{1-\theta} - \theta \log \frac{\theta}{1-\theta} = H(\theta).$$

This recovers Item 4 of Lemma 2.12.

B Interpolating and Encoding the Database

In this section, we prove the claims made in Sections 2.3 and 4.3 about being able to efficiently represent the database $\mathsf{DB} \in \{0,1\}^n$ as a polynomial f_{DB}.

Encoding. We first describe how to encode each index.

Lemma B.1. *Let D, d, m, n be positive integers such that $n \leq \mathsf{Count}(D, d, m)$. Then for any index $i \in [n]$, the encoding $E_{D,d,m}(i)$ can be computed in time $\mathsf{poly}(D, d, m)$.*

Proof. For cleanliness, the algorithm can be viewed as comprising an "offline phase" that does not depend on i, followed by an "online phase" that will depend on i. We begin by detailing the offline phase, wherein we will precompute $\mathsf{Count}(D', d, m')$ for all integers $D' \in [0, D]$ and $m' \in [1, m]$. This is straightforward using dynamic programming: for the base case, we note that $\mathsf{Count}(D', d, 1) = 1$ if $D' \leq d$ and 0 if $D' > d$. For the recursion step, note for any $m' > 1$ that:

$$\mathsf{Count}(D', d, m') = \sum_{r=0}^{\min(d, D')} \mathsf{Count}(D' - r, d, m' - 1).$$

(This follows by iterating and summing over all possible values r of the first entry in the vector of dimension m'.) The runtime of this phase is $O(Ddm)$. As conventions, we will take $\mathsf{Count}(D', d, 0)$ to be 1 if $D' = 0$ and 0 otherwise and we will also take $\mathsf{Count}'(D', d, m')$ to be 0 whenever $D' < 0$.

In the online phase, our main tool is the following function $g(x_1, \dots, x_m)$: for any $\mathbf{x} = (x_1, \dots, x_m) \in S_{\leq D,d,m}$, $g(\mathbf{x})$ is the number of vectors in $S_{D,d,m}$ that are lexicographically strictly less than $\mathbf{x}$. By definition, $g(\mathbf{x})$ is equal to the sum over all indices $j \in [1, m]$ of the number of vectors in $S_{D,d,m}$ that agree with $\mathbf{x}$ on the first $j-1$ entries but are equal to some $x'_j < x_j$ in the jth position. For a particular value of x'_j, this is exactly $\mathsf{Count}(D - \sum_{i=1}^{j-1} x_i - x'_j, d, m-j)$, which we have precomputed in the offline phase. Thus, $g(\mathbf{x})$ can be computed in time $O(dm)$.

Finally, suppose we are given an index i; we wish to find the ith lexicographically smallest vector $\mathbf{x}$ in $S_{D,d,m}$. The observation is this is straightforward to compute $\mathbf{x}$ one entry at a time; we can use $g(\cdot)$ to determine whether we have chosen the correct entry. This is doable with $O(dm)$ calls to $g(\cdot)$, for a total runtime of $O(d^2m^2 + Ddm)$. □

Lemma B.2. *Let D, d, m, n be positive integers such that $n \leq \mathsf{Count}_{\leq}(D, d, m)$. Then for any index $i \in [n]$, the encoding $E_{\leq D,d,m}(i)$ can be computed in time $\mathsf{poly}(D, d, m)$.*

Proof Sketch. The proof is very similar to that of Lemma B.1 so we only sketch the changes here. In the precomputation phase, we additionally precompute $\mathsf{Count}(\leq D', d, m')$ for all $D' \in [0, D]$ and $m' \in [1, m]$. This can be done generically from the existing precomputation of $\mathsf{Count}(D', d, m')$ in $O(Ddm)$ time by just noting that $\mathsf{Count}_{\leq}(D', d, m') = \mathsf{Count}(D', d, m') + \mathsf{Count}_{\leq}(D' - 1, d, m')$ for $D' \geq 1$.

Then in the online phase, we define $g(\mathbf{x})$ similarly except it will now count vectors in $S_{\leq D,d,m}$ by summing $\mathsf{Count}_{\leq}(D - \sum_{i=1}^{j-1} x_i - x'_j, d, m-j)$ over indices j and values $x'_j < x_j$. □

Interpolating. The task of interpolating f_{DB} has been studied in prior work [vdHS12, FKW25] as a special case of "trimmed multipoint evaluation/interpolation". The following theorem is a straightforward corollary of [vdHS12, Theorem 1]:

Theorem B.3. *Assume $d, |\mathbb{F}| = O(1)$. For any $n \leq \mathsf{Count}_{\leq}(D, d, m)$ and target vector $\mathbf{y} \in \mathbb{F}_q^n$, the coefficients of an $f_{\mathsf{DB}} : \mathbb{F}_q^m \to \mathbb{F}_q$ with total degree $\leq D$ and individual degree $\leq d$ such that $f_{\mathsf{DB}}(E_{\leq D,d,m}(i)) = \mathsf{DB}_i \forall i \in [n]$ can be computed in time*

$$O\left(Dm + m^3 \cdot \mathsf{Count}_{\leq}(D, d, m)\right).$$

Proof. Borrowing language from the statement of [vdHS12, Theorem 1], we will set the index set I to be $S_{\leq D,d,m}$ (which is indeed downward-closed) and $V(I, v) = I$. Noting that $d = O(1)$, the term:

$$\frac{\mathsf{M}(d_1)\log d_1}{d_1} + \dots + \frac{\mathsf{M}(d_m)\log d_m}{d_m}$$

is simply $O(m)$. The theorem then tells us that f_{DB} can be interpolated in at most $O(m \cdot \mathsf{Count}_{\leq}(D, d, m))$ field operations, however this ignores costs of indexing into $E_{\leq D,d,m}$. By Lemma B.2, this can be handled with a preprocessing phase that runs in time $O(Dm)$, after which indexing into $E_{\leq D,d,m}$ takes time $O(m^2)$ per query. This gives an overall runtime bound of:

$$O\left(Dm + m^3 \cdot \mathsf{Count}_{\leq}(D, d, m)\right),$$

as desired. □

C Additional Material on Two-Server PIR with Preprocessing

C.1 Proof of Corollary 3.3

Proof of Corollary 3.3. We will make use of the following inequalities, which all hold for any $n > 10^6$, $m > 0$, and $k \in [0, \lfloor m/2 \rfloor]$:

$$\sum_{j=0}^{k} \binom{m}{j} \leq 2^{m \cdot H(k/m)} \tag{6}$$

$$\binom{m}{\lfloor m/2 \rfloor} \geq \frac{2^m}{\sqrt{\pi/2} \cdot \sqrt{m+2}} \tag{7}$$

$$\left\lceil n/2 \cdot \sqrt{\pi/2} \cdot \sqrt{\lceil \log n + \log\log n + 2 \rceil} \right\rceil < 0.75n\sqrt{\log n} \tag{8}$$

$$(1.5n\sqrt{\log n})^{H(1/4)} \cdot \frac{1}{1 - 2^{-H(1/4)}} \leq 12n^{0.82} \tag{9}$$

For a given database size $n > 10^6$, define $K = \lceil \log n + \log\log n \rceil$ and

$$N = 2 \left\lceil n/2 \cdot \sqrt{\pi/2} \cdot \sqrt{K+2} \right\rceil. \tag{10}$$

By Equation (8), we know that $N < 2^{K+1}$ and moreover N is even, so we may use N's binary representation to write $N = 2^{m_1} + 2^{m_2} + \ldots + 2^{m_k}$, for some k and integers $1 \leq m_k < \ldots < m_2 < m_1 \leq K$. Our strategy is to break the n-bit database into k "subdatabases" of size $\binom{m_1}{\lfloor m_1/2 \rfloor}, \ldots, \binom{m_k}{\lfloor m_k/2 \rfloor}$, and apply the construction in Theorem 3.2 with numbers of variables m_i and total degrees $D_i := \lfloor m_i/2 \rfloor$. We first justify that this is sufficient: indeed, we have

$$\begin{aligned}
\sum_{i=1}^{k} \binom{m_i}{\lfloor m_i/2 \rfloor} &\geq \sum_{i=1}^{k} \frac{2^{m_i}}{\sqrt{\pi/2} \cdot \sqrt{m_i+2}} \text{ (Equation (7))} \\
&\geq \frac{\sum_{i=1}^{k} 2^{m_i}}{\sqrt{\pi/2} \cdot \sqrt{K+2}} \\
&= \frac{N}{\sqrt{\pi/2} \cdot \sqrt{K+2}} \\
&\geq n \text{ (Equation (10))}.
\end{aligned}$$

We now check each requirement:

- The server storage is exactly $\sum_{i=1}^{k} 2^{m_i} = N \leq 1.5n\sqrt{\log n}$, using Equation (8).
- On each query, each server accesses and returns the following number of bits:

$$\begin{aligned}
\sum_{i=1}^{k} \sum_{j=0}^{\lfloor m_i/4 \rfloor} \binom{m_i}{j} &\leq \sum_{i=1}^{k} 2^{m_i \cdot H(1/4)} \text{ (Equation (6))} \\
&\leq 2^{m_1 \cdot H(1/4)} \left(1 + 2^{-H(1/4)} + 2^{-2H(1/4)} + \ldots\right) \\
&\leq N^{H(1/4)} \cdot \frac{1}{1 - 2^{-H(1/4)}} \\
&\leq (1.5n\sqrt{\log n})^{H(1/4)} \cdot \frac{1}{1 - 2^{-H(1/4)}} \\
&\leq 12n^{0.82} \text{ (Equation (9))}.
\end{aligned}$$

This completes our proof. □

C.2 Shrinking the Communication with Fully Homomorphic Encryption

It will help us to explicitly state the complexity of the client's reconstruction procedure, viewed as a function of its randomized state.

Definition C.1 (Client polynomial complexity per query)**.** For a number of servers $s \geq 2$ and a randomized state st that consists of field elements in $\mathbb{F}$, we say that the client's polynomial complexity is $(D, k, |\mathsf{st}|)$ if there exist polynomials $g_{\mathsf{ans}_1}, g_{\mathsf{ans}_2}, \ldots, g_{\mathsf{ans}_s}$ over $\mathbb{F}$, each with total degree $\leq D$ and at most k monomials, such that:

$$\mathsf{Reconstruct}(\mathsf{st}, \mathsf{ans}_1, \ldots, \mathsf{ans}_s) = \sum_{i=1}^{s} g_{\mathsf{ans}_i}(\mathsf{st}).$$

We also require that it is possible to write the coefficients of g_{ans} given ans in time $k^{1+o(1)} \cdot \mathsf{poly}(D, \log|\mathsf{st}|, \log|\mathbb{F}|)$.

To combine our information-theoretic schemes with homomorphic encryption, we begin with a lemma about our two-server protocol:

Lemma C.2. *In the scheme in Fig. 3, the client's polynomial complexity per query over $\mathbb{F}_2$ is*

$$\left(\lfloor D/2 \rfloor, n^{H(\theta/2)/H(\theta)+o(1)}, \frac{\log n}{H(\theta)}(1+o(1))\right).$$

Proof. Recall from Fig. 3 that the client's output is:

$$\begin{aligned}
&\sum_{\substack{\mathbf{t} \in S_{\leq \lfloor D/2 \rfloor, 1, m} \\ \mathsf{supp}(\mathbf{t}) \subseteq \mathsf{supp}(\mathbf{p})}} \left(f(\mathsf{qu}_1 + \mathbf{t}) + f(\mathsf{qu}_2 + \mathbf{t})\right) \\
&= \sum_{\mathbf{t} \in S_{\leq \lfloor D/2 \rfloor, 1, m}} \left(f(\mathsf{qu}_1 + \mathbf{t}) + f(\mathsf{qu}_2 + \mathbf{t})\right) \cdot \prod_{i \in \mathsf{supp}(\mathbf{t})} p_i \\
&= \sum_{i=1}^{2} \left[\sum_{\mathbf{t} \in S_{\leq \lfloor D/2 \rfloor, 1, m}} f(\mathsf{qu}_i + \mathbf{t}) \cdot \prod_{i \in \mathsf{supp}(\mathbf{t})} p_i \right].
\end{aligned}$$

This is the sum over $i = 1, 2$ of a polynomial in $\mathbf{p}$ of total degree $\leq \lfloor D/2 \rfloor$ consisting of $\mathsf{Count}_{\leq}(\lfloor D/2 \rfloor, 1, m) = n^{H(\theta/2)/H(\theta)+o(1)}$ monomials, that can be fully specified by the answer ans_i sent by server i. Moreover, the coefficients can easily be written out since they are directly entries of ans_i. The client's state consists of m elements of $\mathbb{F}_2$. The conclusion follows from Equation (2). □

Lemma C.2 tells us that our PIR scheme can be viewed as follows: the client sends query points $\mathsf{qu}_1, \mathsf{qu}_2 \in \mathbb{F}_2^m$ to the two servers, who respond with $\mathsf{ans}_1, \mathsf{ans}_2$. These answers specify polynomials $g_1, g_2 : \mathbb{F}_2^m \to \mathbb{F}_2$, which are each of total degree $\leq \lfloor D/2 \rfloor$ and comprise at most $n^{H(\theta/2)/H(\theta)+o(1)}$ monomials. The client's reconstruction computes $g_1(\mathbf{p}) + g_2(\mathbf{p})$. The goal is to let the servers perform this reconstruction under the hood of homomorphic encryption.

Proof of Theorem 3.9, assuming FHE. We denote the FHE scheme as $(\mathsf{FHE.Gen}, \mathsf{FHE.Enc}, \mathsf{FHE.Eval}, \mathsf{FHE.Dec})$. On security parameter $\lambda \in \mathbb{N}$, we modify Construction 3.1 as follows:

- In the query phase, the client generates a secret key and an evaluation key as $(\mathsf{sk}, \mathsf{ek}) \leftarrow \mathsf{FHE.Gen}(1^\lambda)$. The client then sends server $j \in \{1, 2\}$ the query point qu_j, as well as the evaluation key ek and the ciphertext $\mathsf{ct} := \mathsf{FHE.Enc}(\mathsf{sk}, \mathbf{p})$, which encrypts each bit of the point of interest $\mathbf{p} \in \mathbb{F}_2^m$.

- In the answering phase, server j computes its answer vector ans_j as in the information-theoretic protocol. This answer vector ans_j defines the reconstruction polynomial $g_{\mathsf{ans}_j} : \mathbb{F}_2^m \to \mathbb{F}_2$. Then, server j homomorphically evaluates this polynomial on the encrypted point of interest, computing $\mathsf{ct}_j := \mathsf{FHE.Eval}(\mathsf{ek}, g_j, \mathsf{ct})$. The server sends ciphertext ct_j back to the user.
- In the reconstruction phase, the user computes $\mathsf{FHE.Dec}(\mathsf{sk}, \mathsf{ct}_j)$ for each $j \in \{1, 2\}$ and sums the results to obtain its output.

Correctness holds by the correctness of the FHE scheme and of the information-theoretic PIR scheme. Security follows by a standard hybrid argument: each server only ever sees a FHE encryption of point $\mathbf{p}$, and an independently distributed, uniformly random point in $\mathbb{F}_2^m$. The server storage and preprocessing time are unaffected. The server time per query now consists of two steps:

1. Computing ans_j as in the original protocol, done in time $n^{H(\theta/2)/H(\theta)+o(1)}$ (Theorem 3.2).
2. Evaluating the reconstruction polynomial g_j under the hood of the FHE scheme. By Lemma C.2, the polynomial g_j has total degree $\leq \lfloor D/2 \rfloor$ over $\mathbb{F}_2$ and at most $n^{H(\theta/2)/H(\theta)+o(1)}$ monomials. Thus, it can be computed by an arithmetic circuit over $\mathbb{F}_2$ comprising at most
$$\lfloor D/2 \rfloor \cdot n^{H(\theta/2)/H(\theta)+o(1)} = n^{H(\theta/2)/H(\theta)+o(1)}$$
gates. Using multi-hop FHE [Hal17], the runtime is $n^{H(\theta/2)/H(\theta)+o(1)} \cdot \mathsf{poly}(\lambda)$.

The client time per query is that of encrypting $m = \frac{\log n}{H(\theta)} \cdot (1 + o(1))$ bits, and then decrypting two ciphertexts. □

In fact, the above construction can be seen as an instantiation of a general compiler that shrinks the download communication in any information-theoretic PIR scheme with linear reconstruction (i.e., where the Reconstruct routine outputs a linear function of each server's answers): in such PIR schemes, it is possible to send an encryption of the client's secret randomness to each server, then to have the server perform its slice of the reconstruction routine under the hood of the homomorphic encryption, then to send back the result. The user simply decrypts and sums each server's contribution.

Remark C.3 (What about somewhat homomorphic encryption?). When composing our PIR schemes with fully homomorphic encryption, the evaluated polynomial is of moderate degree: namely, $\lfloor D/2 \rfloor = \Theta(\log n)$. Our protocols are also compatible with somewhat homomorphic encryption schemes, that can only evaluate polynomials of lower degree [BGN05, GHV10, CGHKV25]. The larger the class of supported polynomials, the smaller the PIR scheme's communication.

C.3 Shrinking the Communication with Linearly Homomorphic Encryption

Ingredient 1: succinct single-server batch PIR from LHE. We will make black-box use of the following known theorem:

Theorem C.4 ([KO97, IKOS04]). *Let $\alpha \in (0, 1)$ and $\epsilon > 0$ be real constants, $n \in \mathbb{N}$ the database size, and $\lambda \in \mathbb{N}$ the security parameter. Assuming compact linearly homomorphic encryption over $\mathbb{F}_2$, there is a single-server "batch PIR" protocol—where the user learns n^α entries of the database of their choosing but the server learns nothing about which set of indices the user accessed—achieving the following efficiency guarantees:*

- *server time per query $n \cdot \mathsf{poly}(\lambda)$, and*

- *communication and client time per query* $n^{\alpha+\epsilon} \cdot \mathsf{poly}(\lambda)$.

Proof Sketch. Assuming compact LHE, [KO97] gives a protocol for single-server PIR with a single query that achieves server time per query $n \cdot \mathsf{poly}(\lambda)$ and communication and client time per query $n^{\epsilon} \cdot \mathsf{poly}(\lambda)$. This can then be boosted to batch PIR using the hashing approach outlined in [IKOS04]. (As stated there, we can retain perfect correctness at the expense of an additional $\mathsf{negl}(\lambda)$ security loss.) □

Ingredient 2: writing the database polynomial as a degree-two polynomial. Next, we show that any polynomial of total degree $\leq D$ and individual degree $\leq d$ in m variables can be written as a degree-two polynomial in $\approx \sqrt{\mathsf{Count}_{\leq}(D, d, m)}$ auxiliary variables. More generally, we prove:

Theorem C.5. *Let q be a prime, $d \geq 1$ an integer, and $\alpha \in (0, 1)$ and $\theta \in (0, d/2]$ real constants. For any $m \in \mathbb{N}$, given m variables $X_1, \ldots, X_m \in \mathbb{F}_q$, there exist $M_Y = 2^{m(\alpha \cdot \mathsf{CntExp}(\theta,d)+o(1))}$ auxiliary variables $Y_i \in \mathbb{F}_q$ and $M_Z = 2^{m((1-\alpha)\cdot \mathsf{CntExp}(\theta,d)+o(1))}$ auxiliary variables $Z_i \in \mathbb{F}_q$ such that:*

- *Each Y_i and Z_i is computable from $X_1, \ldots, X_m$ in $\mathsf{poly}(m, \log d, \log q)$ time, and*
- *For any polynomial $f : \mathbb{F}_q^m \to \mathbb{F}_q$ of total degree $\leq \theta m \pm O(1)$ and individual degree $\leq d$, there exists a polynomial $g : \mathbb{F}_q^{M_Y+M_Z} \to \mathbb{F}_q$ of degree ≤ 1 in the Y_i's and ≤ 1 in the Z_i's (i.e., it is bilinear) such that $f(X_1, \ldots, X_m) = g(Y_1, \ldots, Y_{M_Y}, Z_1, \ldots, Z_{M_Z})$.*
- *If f consists of k monomials, then we can construct g in time $k \cdot \mathsf{poly}(m, \log d, \log |\mathbb{F}|)$.*

When $\alpha = 1/2$, this means that we only need $\approx \sqrt{\mathsf{Count}(D, d, m)}$ auxiliary variables.

Proof. Throughout this proof, we will extend indices cyclically so that x_{m+1} denotes x_1, x_{m+2} denotes x_2, and so on. Let $D = \theta m \pm O(1)$ be the total degree of f. Our auxiliary variables can be specified as follow:

- $\{Y_i\}$: all monomials in $x_1, \ldots, x_m$ of total degree $\leq \lfloor \alpha D + (D/m + d/2) \rfloor$ and individual degree $\leq d$ which are supported on some $\lceil \alpha m \rceil$ consecutive variables (where the notion of "consecutiveness" allows for cyclic wrap-around).
- $\{Z_i\}$: all monomials in $x_1, \ldots, x_m$ of total degree $\leq \lfloor (1-\alpha)D + d/2 \rfloor$ and individual degree $\leq d$ which are supported on some $\lceil (1-\alpha)m \rceil$ consecutive variables.

Noting that $D/m + d/2 = O(1)$, it is clear by Theorem A.2 that there are at most M_Y and M_Z of these variables respectively (we will pick up a factor of m because of the choice of which consecutive variables a monomial is supported on, but this will be absorbed by the $2^{o(m)}$). It is also clear that each monomial's value is computable from $X_1, \ldots, X_m$ in time $\mathsf{poly}(m, \log d, \log |\mathbb{F}|)$.

To show the claim, by linearity, it suffices to consider the case where f is a monomial and that in this case we can construct the relevant g in time $\mathsf{poly}(m, \log d, \log |\mathbb{F}|)$. Suppose $f(x_1, \ldots, x_m) = x_1^{a_1} \ldots x_m^{a_m}$, where $a_i \in [0, d]$ for all i and $\sum_{i=1}^m a_i = D'$ for some $D' \leq D$. For each $i \in [m]$, consider the sum $s_i := \sum_{j=0}^{\lceil \alpha m \rceil - 1} a_{i+j}$. Observe that $s_{i+1} - s_i = a_{i+\lceil \alpha m \rceil} - a_i \in [-d, d]$. It follows from that there exists an index i^* such that: $\left| s_{i^*} - \frac{1}{m} \sum_{i \in [m]} s_i \right| \leq d/2$. We hence have:

$$s_{i^*} \in \left[\frac{1}{m} \sum_{i \in [m]} s_i - \frac{d}{2}, \frac{1}{m} \sum_{i \in [m]} s_i + \frac{d}{2} \right]$$

$$= \left[\frac{\lceil \alpha m \rceil}{m} D' - \frac{d}{2}, \frac{\lceil \alpha m \rceil}{m} D' + \frac{d}{2} \right]$$
$$\subseteq \left[\alpha D' - d/2, (\alpha + 1/m) D' + d/2 \right].$$

Now, consider the monomials:

$$Y := \prod_{j=0}^{\lceil \alpha m \rceil - 1} x_{i^*+j}^{a_{i^*+j}} \text{ and } Z := \prod_{j=\lceil \alpha m \rceil}^{m-1} x_{i^*+j}^{a^*+j}.$$

Y and Z are respectively supported on some $\lceil \alpha m \rceil$ and $\lceil (1-\alpha)m \rceil$ consecutive variables and also have individual degree $\leq d$. Moreover, the total degree of Y is $s_{i^*} \leq (\alpha + 1/m)D' + d/2 \leq \alpha D + (D/m + d/2)$, hence Y will be included in the Y_i's. The total degree of Z is:

$$\left(\sum_{i \in [m]} s_i \right) - s_{i^*} \leq D' - (\alpha D' - d/2)$$
$$= (1-\alpha)D' + d/2$$
$$\leq (1-\alpha)D + d/2,$$

hence Z will be included in the Z_i's as well. Multiplying Y and Z together is a bilinear computation that recovers $f(x_1, \ldots, x_m)$. Finally, g is efficiently constructible from f because we need only find i^*, which can in turn be done by calculating all s_i's which is achievable in time $\mathsf{poly}(m, \log d)$. □

Ingredient 3: quasi-uniform index encodings. The main idea in Theorem C.5 is to show that any monomial can be sliced contiguously into two smaller monomials that both have total degree close to the expected total degree. It will be beneficial for us to also ensure that when we encode a database index into a point in $\mathbb{F}_2^m$, that the Hamming weight of this point is roughly uniformly distributed. Formally we show the following:

Lemma C.6 (Compare with Lemma B.1)**.** *Fix a real constant $\theta \in (0, 1/2]$. Let m be the number of variables and $D = \lfloor \theta m \rfloor$. Let $n = 2^{m \cdot H(\theta) + o(m)}$. Then there exists an encoding $E : [n] \to \{0,1\}^m$ satisfying the following properties:*

- *For any index $i \in [n]$, we can compute $E_{D,d,m}(i)$ in time $\mathsf{poly}(D, d, m)$.*
- *For all $i \in [n]$, $E_{D,d,m}(i)$ has Hamming weight exactly D.*
- *For all $i \in [n]$ and $j, l \in [m]$, the Hamming weight of $E_{D,d,m}(i)$ restricted to coordinates $j, j+1, \ldots, j+l-1$ is at most $\theta l + O(\sqrt{m})$.*

Proof. Let $b = \lfloor \sqrt{m} \rfloor$. We can write $m = m_1 + \ldots + m_k$ for $k \leq O(\sqrt{m})$ and $m_i \in \{b, b+1\}$ for all i. We can also select integers $D_1, \ldots, D_k$ such that $D_1 + \ldots + D_k = D$ and $D_i = \theta m_i \pm O(1)$ for all i. Our encoding will consist of vectors $\mathbf{p} = \mathbf{p}_1 \| \mathbf{p}_2 \| \ldots \| \mathbf{p}_k \in \{0,1\}^{m_1} \times \{0,1\}^{m_2} \ldots \times \{0,1\}^{m_k}$ such that $\mathbf{p}_i$ has Hamming weight exactly D_i. Accordingly, segment $[m]$ into intervals $I_1, \ldots, I_k$ of respective lengths $m_1, \ldots, m_k$. We check each requirement in sequence:

- We can compute the ith such vector $\mathbf{p}$ in lexicographic order by essentially making black-box use of Lemma B.1 to find $\mathbf{p}_1, \ldots, \mathbf{p}_k$ individually. Thus we can compute $E_{D,d,m}(i)$ efficiently.

- The Hamming weight of each of these vectors $\mathbf{p}$ is D by construction.
- Consider any contiguous interval of length l of the entries of $\mathbf{p}$. It will consist of some intervals $I_i, I_{i+1}, \ldots, I_{i+j-1}$ in their entirety, as well as potentially part of the intervals I_{i-1} and I_{i+j}.

  The Hamming weight of $\mathbf{p}$ restricted to this interval is hence at most $D_{i-1} + \ldots + D_{i+j} \leq \theta(m_i + \ldots + m_{i+j-1}) + O(j) + 2(b+1) \leq \theta l + O(\sqrt{m})$.
- Finally, the number of such vectors $\mathbf{p}$ is equal to $\prod_{i=1}^{k} \mathsf{Count}(D_i, 1, m_i) = \prod_{i=1}^{k} 2^{m_i \cdot H(\theta) + o(m_i)} = 2^{m \cdot H(\theta) + o(k \max m_i)} = 2^{m \cdot H(\theta) + o(m)}$, so we can accommodate a database of size $n = 2^{m \cdot H(\theta) + o(m)}$.

□

Proof of Theorem 3.9, assuming compact LHE. Let $\alpha = \frac{H(\theta/2)}{\theta + H(\theta/2)}$. We denote the LHE scheme as

$$(\mathsf{LHE.Gen}, \mathsf{LHE.Enc}, \mathsf{LHE.Eval}, \mathsf{LHE.Dec}).$$

We also denote the batch PIR scheme implied by Theorem C.4 as

$$(\mathsf{BatchPIR.Query}, \mathsf{BatchPIR.Answer}, \mathsf{BatchPIR.Reconstruct}).$$

The first modification to the protocol in Construction 3.1 is that our interpolation set will be the set of points used in Lemma C.6. The correctness proofs of our original scheme (e.g. Theorem 3.8) only require any point in the interpolation set to have Hamming weight exactly D, which is ensured by Lemma C.6. Secondly, we can efficiently retrieve the ith among these vectors.

Before detailing further modifications, we summarize the main ideas of our protocol. Let $\mathbf{p} = E_{D,1,m}(i)$ be the client's point of interest. By Lemma C.2 and Theorem C.5, we can write the client's reconstruction as $\mathbf{y}^\top \mathbf{A}_1 \mathbf{z} + \mathbf{y}^\top \mathbf{A}_2 \mathbf{z}$ over $\mathbb{F}_2$ subject to the following constraints:

- $\mathbf{z}$ has dimension $M_Z = 2^{m((1-\alpha)H(\theta/2)+o(1))} = n^{(1-\alpha)H(\theta/2)/H(\theta)+o(1)}$;
- $\mathbf{y}$ has dimension $M_Y = 2^{m(\alpha H(\theta/2)+o(1))} = n^{\alpha H(\theta/2)/H(\theta)+o(1)}$;
- any particular entry of any of the vectors $\mathbf{y}, \mathbf{z}$ can be computed from $\mathbf{p}$ in $\mathsf{poly}(m) = \mathsf{poly}(\log n)$ time; and
- the servers can use their query ($\mathbf{r}$ or $\mathbf{r} + \mathbf{p}$) to reconstruct their corresponding $\mathbf{A}$ matrix in time $n^{H(\theta/2)/H(\theta)+o(1)}$.

The final observation we will need is that many entries of $\mathbf{y}$ are in fact zero. To see this, recall from the proof of Theorem C.5 that the entries of $\mathbf{y}$ are multilinear monomials in the entries of $\mathbf{p}$ of total degree $\leq \theta\alpha m/2 + O(1)$ that are supported on some $\lceil \alpha m \rceil$ consecutive variables. By our choice of encoding in Lemma C.6, the Hamming weight of $\mathbf{p}$ restricted to any $\lceil \alpha m \rceil$ consecutive variables is at most $\theta\alpha m + O(\sqrt{m})$. Hence the number of these monomials that will be nonzero is at most $2^{\theta\alpha m + O(\sqrt{m}) + o(m)} = 2^{\theta\alpha m + o(m)}$. More strongly, the client can calculate a compact representation of $\mathbf{y}$—namely, the list of indices where it is 1—in time $2^{\theta\alpha m + o(m)} = n^{\alpha\theta/H(\theta)+o(1)}$. (This is because they can quickly find the indices in the support of $\mathbf{p}$, then iterate through all subsets of size $\leq \theta\alpha m/2 + O(1)$ supported on some $\lceil \alpha m \rceil$ consecutive indices. For each subset, the relevant index can be quickly found using a procedure analogous to that in Lemma B.1.) Bearing these in mind, we can now complete our presentation of the protocol. We will make reference to its information-theoretic variant that we presented in Construction 3.1.

- In the query phase, the client generates a secret key and an evaluation key as $(\mathsf{sk}, \mathsf{ek}) \leftarrow \mathsf{LHE.Gen}(1^\lambda)$. The client computes the auxiliary variables $\mathbf{z} \in \mathbb{F}_2^{M_Z}$ corresponding to the point of interest $\mathbf{p}$, as well as the list $L \subseteq [M_Y]$ of indices where $\mathbf{y}$ is 1.

  To server j for $j \in \{1, 2\}$, the client sends its original query qu_j along with the evaluation key ek, the ciphertext $\mathsf{ct} := \mathsf{LHE.Enc}(\mathsf{sk}, \mathbf{z})$, and the query $\mathsf{qu}_{\mathsf{batch}} := \mathsf{BatchPIR.Query}(L)$. Here and in the following, we will use the batch PIR from Theorem C.4 with database size M_Y.

- In the answering phase, server j computes ans_j as in the information-theoretic protocol, and then computes the matrix $\mathbf{A}_j \in \mathbb{F}_2^{M_Y \times M_Z}$. The server then evaluates $\mathsf{ct}_j := \mathsf{LHE.Eval}(\mathsf{ek}, \mathbf{x} \mapsto \mathbf{A}_j\mathbf{x}, \mathsf{ct})$. ct_j is an LHE ciphertext of $\mathbf{A}_j\mathbf{z}$ and can hence be thought of as a $M_Y \times \mathsf{poly}(\lambda)$ matrix of bits.

  For each of the $\mathsf{poly}(\lambda)$ columns $\mathsf{col} \in \{0,1\}^{M_Y}$ of ct_j, the server sends back $\mathsf{BatchPIR.Answer}(\mathsf{col}, \mathsf{qu}_{\mathsf{batch}})$ i.e., it will answer the client's BatchPIR query for the smaller database col and indices L.

- In the reconstruction phase, the client first uses BatchPIR.Reconstruct to recover all rows of $\mathsf{ct}_1, \mathsf{ct}_2$ corresponding to nonzero entries of $\mathbf{y}$. They then use LHE.Dec in each row to decrypt and recover the corresponding entries of $\mathbf{A}_1\mathbf{z} + \mathbf{A}_2\mathbf{z}$. Since these are all the entries where $\mathbf{y}$ is 1, the client can finish by simply taking the XOR of the $|\mathsf{supp}(\mathbf{y})|$ bits it recovered.

Correctness and security are immediate, and the server storage and preprocessing time are unchanged. We address the other efficiency metrics below:

- Server time per query: to compute ans_j, the time is $n^{H(\theta/2)/H(\theta)+o(1)}$. To compute the matrix $\mathbf{A}_j$, the runtime is the same. Homomorphic evaluation (assuming it is multi-hop) takes the same amount of time times $\mathsf{poly}(\lambda)$, since to evaluate $\mathbf{x} \mapsto \mathbf{A}_j\mathbf{x}$ we require $O(M_Y M_Z) = n^{H(\theta/2)/H(\theta)+o(1)}$ arithmetic operations over $\mathbb{F}_2$. Finally, each of the $\mathsf{poly}(\lambda)$ calls to BatchPIR.Answer can be run in time $M_Y \cdot \mathsf{poly}(\lambda)$, which is dominated by the previous terms.

- Client time per query (which for us is identical to the communication up to $n^{o(1)} \cdot \mathsf{poly}(\lambda)$ factors):

  - Computing and encrypting $\mathbf{z}$ can be done in time
  $$M_Z \cdot \mathsf{poly}(\log n, \lambda) = n^{(1-\alpha)H(\theta/2)/H(\theta)+o(1)} \cdot \mathsf{poly}(\lambda) = n^{\frac{\theta \cdot H(\theta/2)}{H(\theta)(\theta+H(\theta/2))}+o(1)} \cdot \mathsf{poly}(\lambda).$$
  - Computing L, generating $\mathsf{BatchPIR.Query}(L)$, and making all the $\mathsf{poly}(\lambda)$ calls to BatchPIR.Reconstruct can be done in time
  $$n^{\alpha\theta/H(\theta)+\epsilon} \cdot \mathsf{poly}(\lambda) = n^{\frac{\theta \cdot H(\theta/2)}{H(\theta)(\theta+H(\theta/2))}+\epsilon} \cdot \mathsf{poly}(\lambda).$$
  - There are $|\mathsf{supp}(\mathbf{y})|$ many calls to LHE.Dec, so the total runtime of these is $n^{\alpha\theta/H(\theta)+o(1)} \cdot \mathsf{poly}(\lambda)$.

This completes the proof of the theorem. □

D Additional Material on Many-Server PIR with Preprocessing

We begin with a remark on the PIR framework given in Section 5.1:

Remark D.1 (Reducing the communication using homomorphic encryption). As in the two-server setting, we can shrink communication by composing this PIR framework with homomorphic encryption, at the cost of settling for computational privacy. It suffices to encrypt the random curve $\mathbf{L}$ along which the query points lie under a fully-homomorphic-encryption scheme [RAD78, Gen09, BV11]. Then, each server can run a portion of the user's reconstruction under the hood of the encryption scheme. If the user's reconstruction time is $n^{1-\Omega(1)}$, as in both constructions we describe, the scheme retains its double efficiency.

D.1 Background on Secret Sharing, Hasse Derivatives, and Hermite Interpolation

Security of Secret Sharing.

Lemma D.2 (Security of secret sharing)**.** *Let t be a positive integer and let $\mathbb{F}$ be a field with $|\mathbb{F}| \geq t$. For any subset $T \subseteq \mathbb{F}$ of size t and any secret $\mathbf{s} \in \mathbb{F}^m$, define the distribution $\mathcal{D}_{\mathbf{s},T}$ as:*

1. *Sample a uniformly random curve $\mathbf{L} : \mathbb{F} \to \mathbb{F}^m$ of degree $\leq t$ such that the coefficient of X^t is vector $\mathbf{s}$.*
2. *Output $\{\mathbf{L}(x) : x \in T\}$.*

Then, $\mathcal{D}_{\mathbf{s},T}$ is uniformly distributed in $(\mathbb{F}^m)^t$.

Proof. Let $T = \{\alpha_1, \ldots, \alpha_t\}$, and let $\mathbf{M}$ be the Vandermonde matrix

$$\begin{bmatrix} 1 & \alpha_1 & \ldots & \alpha_1^t \\ 1 & \alpha_2 & \ldots & \alpha_2^t \\ \vdots & \vdots & \ddots & \vdots \\ 1 & \alpha_t & \ldots & \alpha_t^t \end{bmatrix} \in \mathbb{F}^{t \times (t+1)}.$$

We will write this as $\begin{bmatrix} \mathbf{M}' & \mathbf{w} \end{bmatrix}$, where $\mathbf{M}' \in \mathbb{F}^{t \times t}$ and $\mathbf{w} \in \mathbb{F}^t$. Then $\mathcal{D}_{\mathbf{s},T}$ can be characterized as follows: sample $\mathbf{v}_1, \ldots, \mathbf{v}_m \leftarrow \mathbb{F}^t$ and output

$$\mathbf{M} \begin{bmatrix} \mathbf{v}_1 & \ldots & \mathbf{v}_m \\ s_1 & \ldots & s_m \end{bmatrix} = \mathbf{M}' \begin{bmatrix} \mathbf{v}_1 & \ldots & \mathbf{v}_m \end{bmatrix} + \mathbf{w}\mathbf{s}^\top.$$

Note that $\mathbf{M}'$ is a $t \times t$ Vandermonde matrix with distinct values $\alpha_1, \ldots, \alpha_t$, thus it is invertible. Since the matrix $\begin{bmatrix} \mathbf{v}_1 & \ldots & \mathbf{v}_m \end{bmatrix}$ is uniformly distributed in $\mathbb{F}^{t \times m}$, it follows that $\mathbf{M}' \begin{bmatrix} \mathbf{v}_1 & \ldots & \mathbf{v}_m \end{bmatrix}$ is also uniform, which implies the conclusion. □

Background on Hasse Derivatives and Interpolation. We follow the exposition in [BGKM22, Section 4] and [BGG+24, Sections 3.2 and 3.3]. Proofs of all statements can be found in [For14, Appendix C].

Let $\mathbb{F}$ be an arbitrary field. For a polynomial $f \in \mathbb{F}[X_1, \ldots, X_m]$ and a vector $\mathbf{e} = (e_1, \ldots, e_m) \in \mathbb{N}^m$, we write $\mathsf{Coeff}_{\mathbf{e}}(f) \in \mathbb{F}$ to denote the coefficient in front of the monomial $\mathbf{X}^{\mathbf{e}} := \prod_{i=1}^m X_i^{e_i}$ in f. Then, the partial derivative of the polynomial $f \in \mathbb{F}[X_1, \ldots, X_m]$ with respect to the vector $\mathbf{e}$ is defined as:

$$\begin{aligned} \partial_{\mathbf{e}} f &= \frac{\partial^{e_1 + \ldots + e_m} f}{\partial X_1^{e_1} \ldots \partial X_m^{e_m}} \\ &= \sum_{\substack{\mathbf{a} \in \mathbb{N}^m \\ \mathbf{a} \geq \mathbf{e}}} \left(\prod_{i=1}^m a_i (a_i - 1) \ldots (a_i - e_i + 1) \right) \cdot \mathsf{Coeff}_{\mathbf{a}}(f) \cdot \mathbf{X}^{\mathbf{a}-\mathbf{e}}. \end{aligned}$$

Throughout this section, when f is univariate (i.e., $m = 1$), we sometimes abuse notation and let $\overline{\partial}_k f$ denote the kth-order Hasse derivative of f for $k \in \mathbb{N}$.

Proposition D.3 (Properties of Hasse Derivatives)**.** For any polynomial $f \in \mathbb{F}[X_1, \ldots, X_m]$ and vector $\mathbf{e} \in \mathbb{N}^m$, the following are true:

- *From Hasse derivatives to partial derivatives:* $\partial_{\mathbf{e}} f = \left(\prod_{i \in [m]} e_i! \right) \cdot \overline{\partial}_{\mathbf{e}} f$.

- *From evaluations to derivatives:* $\overline{\partial}_{\mathbf{e}} f$ is the coefficient of $\mathbf{Z}^{\mathbf{e}}$ in $f(\mathbf{X}+\mathbf{Z})$, viewed as a polynomial in $\mathbf{Z} = (Z_1, \ldots, Z_m)$.
- *Derivatives of derivatives:* For vectors $\mathbf{e}_1, \mathbf{e}_2 \in \mathbb{N}^m$, we have

$$\overline{\partial}_{\mathbf{e}_1} \overline{\partial}_{\mathbf{e}_2} f = \binom{\mathbf{e}_1 + \mathbf{e}_2}{\mathbf{e}_1} \overline{\partial}_{\mathbf{e}_1+\mathbf{e}_2} f.$$

  In particular, if $\mathbf{e}_1$ and $\mathbf{e}_2$ have disjoint support, then $\overline{\partial}_{\mathbf{e}_1} \overline{\partial}_{\mathbf{e}_2} f = \overline{\partial}_{\mathbf{e}_1+\mathbf{e}_2} f$.

The following statements use Hasse derivatives for interpolation, and show how to evaluate them efficiently.

Lemma D.4 (Hermite Interpolation [vzGG03, Section 5.6])**.** *For any univariate polynomial $f \in \mathbb{F}[X]$ of degree at most $d \in \mathbb{N}$, let $k \geq 1$ be an integer that is less than or equal to the number of field elements in $\mathbb{F}$. Let $\alpha_1, \ldots, \alpha_k$ be k distinct elements from $\mathbb{F}$ and let $e_1, \ldots, e_k$ be positive integers such that $e_1 + \ldots + e_k > d$. Then, it is possible to reconstruct the polynomial f from the evaluations $\left\{ \overline{\partial}_{j-1} f(\alpha_i) : i \in [k], j \in [e_i] \right\}$, in $O(d^2)$ operations over $\mathbb{F}$.*

Lemma D.5 (Chain Rule for Hasse Derivatives)**.** *Let $f \in \mathbb{F}[X_1, \ldots, X_m]$ be a polynomial with total degree $\leq D$ and individual degree $\leq d$ and let $\mathbf{g} = (g_1, \ldots, g_m) \in \mathbb{F}[X]^m$ be a polynomial curve of degree at most u. Let $h = f \circ \mathbf{g} \in \mathbb{F}[X]$. Then, for any $k \in \mathbb{N}$, it holds that:*

$$\overline{\partial}_k h(X) = \sum_{\substack{\mathbf{e} \in \{0,1,\ldots,d\}^m : \\ \|\mathbf{e}\|_1 \leq k}} \overline{\partial}_{\mathbf{e}} f(\mathbf{g}(X)) \cdot \mathsf{Coeff}_{Y^k} \left(\prod_{j=1}^{m} \left(g_j(X+Y) - g_j(X) \right)^{e_j} \right).$$

Moreover, given the curve $\mathbf{g}$ and a point $x \in \mathbb{F}$, it is possible to compute

$$\left\{ \mathsf{Coeff}_{Y^k} \left(\prod_{j=1}^{m} \left(g_j(x+Y) - g_j(x) \right)^{e_j} \right) : \mathbf{e} \in \{0,1,\ldots,d\}^m \text{ s.t. } \|\mathbf{e}\|_1 \leq k \right\}$$

in time $\mathsf{Count}_{\leq}(k,d,m) \cdot \mathsf{poly}(k, u, \log|\mathbb{F}|)$.

Proof. The first claim follows immediately from [BGKM22, Lemma 6.2], together with the fact that the polynomial f has individual degree $\leq d$ (so any Hasse derivative of order $> d$ along any axis must vanish). For the second claim, we write $g_j(t) = \sum_{i=0}^{u} a_{ji} t^i$. Then, we have:

$$g_j(t+z) - g_j(t) = \sum_{i=1}^{u} a_{ji}((t+z)^i - t^i) = \sum_{l=1}^{u} \left(\sum_{i=l}^{u} a_{ji} \binom{i}{l} t^{i-l} \right) z^l.$$

We can compute all these coefficients in time $\mathsf{poly}(u, \log|\mathbb{F}|)$. Now, for a particular vector $\mathbf{e}$, computing $\prod_{j=1}^{m} \left(g_j(t+z) - g_j(t) \right)^{e_j}$ amounts to multiplying $\|\mathbf{e}\|_1 \leq k$ univariate polynomials, each of degree $\leq u$. This can be done in time $\mathsf{poly}(k, u, \log|\mathbb{F}|)$. Finally, the number of vectors $\mathbf{e}$ that we need to iterate over is $\mathsf{Count}_{\leq}(k,d,m)$. Multiplying these costs completes the proof. □

Lemma D.6 (Full-domain evaluation of a polynomial and its Hasse Derivatives)**.** *For a prime q, let $f \in \mathbb{F}_q[X_1, \ldots, X_m]$ be a polynomial of individual degree $\leq d$. Let $\mathcal{E}$ denote a collection of vectors in $\mathbb{N}^m$, such that all entries of each vector in $\mathcal{E}$ are at most d. Then, there exists an algorithm that computes the set of evaluations $\left\{ \overline{\partial}_{\mathbf{e}} f(\mathbf{x}) : \mathbf{x} \in \mathbb{F}_q^m, \mathbf{e} \in \mathcal{E} \right\}$. Moreover, this algorithm runs in time $O\left(q^m \cdot |\mathcal{E}| \cdot m \cdot \mathsf{poly}(\log q)\right)$.*

Construction D.7 (Many-Server PIR with Preprocessing)**.** Parameterized by number of servers $s \geq 2$, number of colluding servers $1 \leq t \leq s-1$, database size n, prime field size $q \geq s$, total degree $D = D(n) \geq 0$, individual degree $1 \leq d < q$, and number of variables $m = m(n) \geq 1$, such that $\mathsf{Count}_{\leq}(D, d, m) \geq n$. Define $E_{\leq D,d,m}(\cdot)$ and $\mathsf{Interpolate}(\cdot)$ as in Section 2.3. As shorthand, we set $\mathcal{L} = \{0, 1, \ldots, D\}$.

$\mathsf{Preprocess}(\mathsf{DB} \in \{0,1\}^n) \to \mathsf{Table}$

- Let $f_{\mathsf{DB}} = \mathsf{Interpolate}(\mathsf{DB}, \leq D, d, m, q)$.
- Slice f_{DB} into polynomials $f_0, \ldots, f_D$ such that f_ℓ is homogeneous and of total degree ℓ, for $\ell \in \mathcal{L}$.
- Build the table $\mathsf{Table} : \mathcal{L} \times \mathbb{F}_q^m \to \mathbb{F}_q$ that, for $\ell \in \mathcal{L}$ and each point $\mathbf{x} \in \mathbb{F}_q^m$, stores the evaluation $f_\ell(\mathbf{x}) \in \mathbb{F}_q$.

$\mathsf{Query}(i \in [n]) \to \left(\mathsf{st}, \mathsf{qu}_1, \ldots, \mathsf{qu}_s\right)$

- Compute $\mathbf{p} = E_{\leq D,d,m}(i) \in S_{D,d,m}$.
- Sample a random curve $\mathbf{L} : \mathbb{F}_q \to \mathbb{F}_q^m$ of degree t with leading coefficient $\mathbf{p}$.
- Output $\mathsf{st} = \mathbf{L}$, $\mathsf{qu}_j = \mathbf{L}(j)$ for $j \in [s]$.

$\mathsf{Answer}^{\mathsf{Table}}(\mathsf{qu} \in \mathbb{F}_q^m) \to \mathsf{ans}$

- Initialize $\mathsf{ans} = \{\}$.
- For $\ell \in \mathcal{L}$ and $\mathbf{e} \in \mathsf{HW}_{\lfloor \ell t/s \rfloor, d, m}$, append $f_\ell(\mathsf{qu} + \mathbf{e})$ to ans, where $f_\ell(\mathsf{qu} + \mathbf{e})$ is retrieved by looking up the corresponding entry in Table.
- Output ans.

$\mathsf{Reconstruct}(\mathsf{st}, \mathsf{ans}_1, \ldots, \mathsf{ans}_s) \to b$

- For $\ell \in \mathcal{L}$, $k \in \{0, 1, \ldots, \lfloor \ell t/s \rfloor\}$, and $j \in [s]$, compute $\overline{\partial}_k h_\ell(j)$ from ans_j, where $h_\ell = f_\ell \circ \mathbf{L}$, via finite differences (Lemma 5.1 and Lemma D.5).
- For each $\ell \in \mathcal{L}$, reconstruct h_ℓ via Hermite interpolation (Lemma D.4).
- Writing each h_ℓ for $\ell \in \mathcal{L}$ as a polynomial in the formal variable X, output $\sum_{\ell=0}^{D} \mathsf{Coeff}_{X^\ell}(h_\ell) \bmod 2$.

Figure 5: Our s-server, t-private PIR with preprocessing protocol.

A useful special case of this lemma is when $\mathcal{E}$ consists of just the zero vector, which gives an algorithm for full-domain evaluation of f.

Proof. Without loss of generality, assume $d \leq q-1$. The algorithm follows from combining [BGKM22, Algorithm 1] and [KU11, Theorem 4.1]. In more detail, we proceed as follows: for each vector $\mathbf{e} \in \mathcal{E}$, we use [BGKM22, Algorithm 1] to compute all coefficients of the Hasse derivative $\overline{\partial}_{\mathbf{e}} f$ in time $d^2 \cdot \mathsf{poly}(\log q) + d^m \cdot m \cdot \mathsf{poly}(\log q) \leq q^m \cdot m \cdot \mathsf{poly}(\log q)$. Then, we plug the resulting polynomial into [KU11, Theorem 4.1] to compute and output its evaluations at all points in $\mathbb{F}^m$ in time $q^m \cdot m \cdot \mathsf{poly}(\log q)$. □

D.2 Additional Material on Construction Optimizing Server Storage

We give the PIR scheme that proves Theorem 5.3 in Fig. 5.
We start with a useful lemma and its proof:

Lemma D.8. *For any prime q and polynomial $f \in \mathbb{F}_q[X_1, \ldots, X_m]$ of individual degree $d \in [q-1]$, let $\alpha_0, \ldots, \alpha_d$ be fixed, distinct elements of $\mathbb{F}_q$. For any $v \in \{0, 1, \ldots, d\}$ and $j \in [m]$, let $\mathbf{v} = v \cdot \mathbf{e}_j \in \mathbb{F}_q^m$ be the vector with value v at position j and 0 elsewhere. Then, there exist constants $c_0, \ldots, c_d \in \mathbb{F}_q$, which depend on the values v and α_i for $i \in \{0, 1, \ldots, d\}$, so that for any point $\mathbf{x} \in \mathbb{F}_q^m$,*

$$\overline{\partial}_{\mathbf{v}} f(\mathbf{x}) = \sum_{i=0}^{d} c_i \cdot f(\mathbf{x} + \alpha_i \mathbf{e}_j).$$

These constants can all be computed in $\mathsf{poly}(d, \log q)$ *time.*

Proof. Define the curve $\mathbf{g} : \mathbb{F}_q \to \mathbb{F}_q^m$ as $\mathbf{g}(X) = \mathbf{x} + \mathbf{e}_j \cdot X$. Define the polynomial $h : \mathbb{F}_q \to \mathbb{F}_q$ as $h = f \circ \mathbf{g}$. Because the polynomial f has individual degree $\leq d$ and the curve $\mathbf{g}$ has individual degree one, the polynomial h also has individual degree $\leq d$. By Lagrange interpolation, there exist coefficients $b_{i,\ell} \in \mathbb{F}_q$ for $i, \ell \in \{0, 1, \ldots, d\}$, that depend only on the values $\alpha_0, \ldots, \alpha_d$, such that:

$$h(X) = \sum_{\ell=0}^{d} \left(\sum_{i=0}^{d} b_{i,\ell} \cdot h(\alpha_i) \right) \cdot X^\ell = \sum_{\ell=0}^{d} \left(\sum_{i=0}^{d} b_{i,\ell} \cdot f(\mathbf{x} + \alpha_i \mathbf{e}_j) \right) \cdot X^\ell.$$

Now, we calculate $\overline{\partial}_v h(0)$ in two ways. On the one hand, from Definition 4.2:

$$\overline{\partial}_v h(0) = \mathsf{Coeff}_v(h) = \sum_{i=0}^{d} b_{i,v} \cdot f(\mathbf{x} + \alpha_i \mathbf{e}_j).$$

On the other hand, applying the chain rule (Lemma D.5, noting that the only term in the sum that does not vanish is where $\mathbf{e} = v\mathbf{e}_j$) gives that:

$$\overline{\partial}_v h(0) = \overline{\partial}_{\mathbf{v}} f(\mathbf{g}(0)) = \overline{\partial}_{\mathbf{v}} f(\mathbf{x}).$$

Equating these two quantities completes the proof. To compute these coefficients efficiently, we note that the coefficients $b_{i,\ell}$ are the entries of the inverse of a $(d+1) \times (d+1)$ Vandermonde matrix (with respect to field elements $\alpha_0, \ldots, \alpha_d$), computable in $\mathsf{poly}(d, \log q)$ field operations. □

Proof of Lemma 5.1.

Proof. Our algorithm will use dynamic programming to compute $\left\{ \overline{\partial}_{\mathbf{a}} f(\mathbf{x} + \mathbf{r}) \right\}$ for all integers $j \in [0, m]$ and vectors $\mathbf{r} \in \{0, 1, \ldots, d\}^m$ and $\mathbf{a} \in S_{\leq \ell, d, m}$ subject to the following constraints:

- $a_i = 0$ for all $i > j$;
- $r_i = 0$ for all $i \leq j$; and
- $\|\mathbf{a}\|_1 + \|\mathbf{r}\|_0 \leq \ell$.

When $j = 0$, these values are those given to the algorithm as input; when $j = m$, these are the desired outputs of the algorithm. So consider some $j > 0$, $\mathbf{a}$, and $\mathbf{r}$ and assume we have already computed these evaluations for $j - 1$. The only evaluations that we have not already computed are those for which $a_j \neq 0$. We may write $\mathbf{a} = \mathbf{a}' + a_j \mathbf{e}_j$, once again letting $\mathbf{e}_j$ denotes the jth standard basis vector. By Proposition D.3 applied to $(\alpha_0, \alpha_1, \ldots, \alpha_d) = (0, 1, \ldots, d)$, we know that:

$$\overline{\partial}_{\mathbf{a}} f(\mathbf{x} + \mathbf{r}) = \overline{\partial}_{a_j \mathbf{e}_j} \left[\overline{\partial}_{\mathbf{a}'} f(\mathbf{x} + \mathbf{r}) \right] = \sum_{i=0}^{d} c_i \cdot \overline{\partial}_{\mathbf{a}'} f(\mathbf{x} + \mathbf{r} + i\mathbf{e}_j),$$

where the scalars $c_0, \ldots, c_d \in \mathbb{F}_q$ come from Lemma D.8 and can depend on a_j. We now claim that for each i, $\overline{\partial}_{\mathbf{a}'} f(\mathbf{x} + \mathbf{r} + i\mathbf{e}_j)$ has already been computed in "layer" $j - 1$. Indeed, $\mathbf{a}'$ is obtained by taking $\mathbf{a}$ (which is zero in indices $> j$) and also zeroing out index j. Secondly, $\mathbf{r} + i\mathbf{e}_j$ is obtained by taking $\mathbf{r}$ (which is zero in all indices $\leq j$) and potentially making index j nonzero; thus all indices $\leq j - 1$ are still zero. Moreover, we are changing entry j to i, thus all entries of $\mathbf{r} + i\mathbf{e}_j$ are still in $\{0, 1, \ldots, d\}$. Finally, we have $\|\mathbf{a}'\|_1 = \|\mathbf{a}\|_1 - a_j$ and $\|\mathbf{r} + \alpha_i \mathbf{e}_j\|_0 \leq \|\mathbf{r}\|_0 + 1$ so it follows that $\|\mathbf{a}'\|_1 + \|\mathbf{r} + i\mathbf{e}_j\|_0 \leq \|\mathbf{a}\|_1 + \|\mathbf{r}\|_0 \leq \ell$ (recalling that $a_j \neq 0$).

Thus each value in "layer" j can be obtained from those in layer $j-1$ with a linear computation whose coefficients can be computed in time $\mathsf{poly}(d, \log q)$ (Lemma D.8). It remains to show the claimed runtime. Up to a $\mathsf{poly}(d, \log q)$ factor, this is equal to the number of evaluations we compute throughout the dynamic programming procedure. To upper bound the number of such pairs $(\mathbf{a}, \mathbf{r})$, consider the vector $\mathbf{b} := \mathbf{a} + \mathbf{r}$. It has entries in $\{0, 1, \ldots, d\}$ and Hamming weight $\leq \ell$ (noting that $\|\mathbf{a}\|_0 \leq \|\mathbf{a}\|_1$). Thus, there are at most $\left|\mathsf{HW}_{\min(\ell,m),d,m}\right|$ choices of $\mathbf{b}$. Moreover, for any particular $\mathbf{b}$, there are at most $m+1$ ways to split it into $\mathbf{a}$ and $\mathbf{r}$. The conclusion follows. □

Lemma D.9. *Let m be a positive integer, $\alpha > 0$ some real constant, and $d \geq 1$ a positive integer. Then,*

$$\max_{w \leq \alpha m} \binom{m}{w} d^w = \begin{cases} 2^{(H(\alpha)+\alpha \log d + o(1))m}, & \text{if } \alpha \leq d/(d+1) \\ (d+1)^{m+o(m)}, & \text{otherwise.} \end{cases}$$

Proof. By an argument analogous to that used in Lemma A.7, it suffices to show

$$\max_{\gamma \in [0,\alpha]} (H(\gamma) + \gamma \log d) = \begin{cases} H(\alpha) + \alpha \log d, & \text{if } \alpha \leq d/(d+1) \\ \log(d+1), & \text{otherwise.} \end{cases}$$

$H(\gamma) + \gamma \log d$ is increasing on $[0, d/(d+1)]$ and decreasing on $[d/(d+1), 1]$. Finally, $\alpha = d/(d+1)$, so we have $H(\alpha) + \alpha \log d = \log(d+1)$. □

Proof of Theorem 5.3

Proof. *Security.* Information theoretic t-privacy follows from the security of the underlying secret-sharing scheme, proved in Lemma D.2.

Correctness. Let $\mathbf{p}$ be the user's point of interest, let $\mathbf{L}$ be the random curve sampled by the user during the query phase, and define the polynomial $h_\ell = f_\ell \circ \mathbf{L}$ for each $\ell \in \{0, 1, \ldots, D\}$. Correctness follows by:

1. *Applying Lemma 5.1.* The lemma on finite differences states that each $\overline{\partial}_{\mathbf{e}} f_\ell(\mathsf{qu}_j)$ for $\ell \in \{0, 1, \ldots, D\}$, $j \in [s]$, and $\mathbf{e} \in S_{\leq \lfloor \ell t/s \rfloor, d, m}$ can be reconstructed linearly (with coefficients depending on $\mathbf{e}$) from $\left\{ f_\ell(\mathsf{qu}_j + \mathbf{r}) \right\}$, where $\mathbf{r}$ ranges over $\mathsf{HW}_{\lfloor \ell t/s \rfloor, d, m}$, which are all included in server j's answer.

2. *Applying Lemma D.5.* The Hasse derivative chain rule lets the user compute $\overline{\partial}_k h_\ell(j)$ for $\ell \in \{0, 1, \ldots, D\}$, for $k \in \{0, 1, \ldots, \lfloor \ell t/s \rfloor\}$, and for $j \in [s]$.

3. *Applying Lemma D.4.* For each $\ell \in \{0, 1, \ldots, D\}$, Hermite interpolation with the polynomial $h_\ell = f_\ell \circ \mathbf{L}$ (which has degree $\leq \deg f_\ell \cdot \deg \mathbf{L} \leq \ell t < s \cdot (\lfloor \ell t/s \rfloor + 1)$) and points $0, 1, \ldots, s-1 \in \mathbb{F}$ recovers the polynomial h_ℓ.

4. *Applying Lemma 4.1.* For every $\ell \in \{0, 1, \ldots, D\}$, write the polynomial h_ℓ in terms of formal variable X. Then, $f_\ell(\mathbf{p})$ is the coefficient $\mathsf{Coeff}_{X^{\ell t}}(h_\ell)$. Finally, the reconstructed bit will be $\sum_{\ell=0}^{D} f_\ell(\mathbf{p})$.

Efficiency. For interpolation, we must set parameters so that $\mathsf{Count}_{\leq}(D, d, m) \geq n$. By Theorem A.2, it suffices to take

$$m = \frac{\log n}{\mathsf{CntExp}_{\leq}(\theta, d)}(1 + o(1)). \tag{11}$$

This parameter setting gives that:

- Server storage and preprocessing time: the server storage is $q^m \cdot (D+1) \cdot \log q$. The preprocessing time consists of (1) writing out the coefficients of f_{DB}; and (2) precomputing all evaluations of f_ℓ for all $\ell \in \{0, 1, 2, \ldots, D\}$. The first of these steps is doable in time $\leq \mathsf{Count}_{\leq}(D, d, m) \cdot \mathsf{poly}(d, m, \log q)$ (by Theorem B.3), and the second step is doable in time $q^m \cdot \mathsf{poly}(d, m, \log q)$ (by Lemma D.6). The latter cost dominates since $d + 1 \leq q$.

  Substituting in Equation (11) implies that this will be $n^{\log q / \mathsf{CntExp}_{\leq}(\theta,d)+o(1)}$.

- Server runtime per query: this is bounded above by $O\left(D \cdot |\mathsf{HW}_{\lfloor Dt/s \rfloor,d,m}|\right)$. Enumerating over all $\leq D$ possible Hamming weights of a vector in $\mathsf{HW}_{\lfloor Dt/s \rfloor,d,m}$, this is bounded above by

$$\mathsf{poly}(D) \cdot \max_{w \leq Dt/s} \binom{m}{w} d^w.$$

  When $\boxed{\theta t/s \leq d/(d+1)}$, by Lemma D.9 this simplifies to:

$$2^{(H(\theta t/s)+\theta t \log d/s+o(1))m} = n^{\frac{s \cdot H(\theta t/s)+\theta t \log d}{s \cdot \mathsf{CntExp}_{\leq}(\theta,d)}+o(1)}.$$

- Client runtime per query: we restrict attention to a particular ℓ and focus on the computation corresponding to one $f := f_\ell$. At the end, all our costs will incur a multiplicative overhead of $D + 1 = n^{o(1)}$.

  We first address costs outside of the application of Corollary 5.1. By Lemma D.4, the runtime of the final Hermite interpolation step is $\mathsf{poly}(D, t, \log q) = n^{o(1)}$.

  The runtime of the Hasse derivative chain rule step (Lemma D.5) is at most:

$$\mathsf{Count}_{\leq}(\lfloor Dt/s \rfloor, d, m) \cdot \mathsf{poly}(Dt/s, t, \log q) = n^{\frac{\mathsf{CntExp}(\theta t/s,d)}{\mathsf{CntExp}_{\leq}(\theta,d)}+o(1)}.$$

  Note that this will be subsumed by the runtime of Corollary 5.1 (up to $n^{o(1)}$ factors) since the output size of applying the corollary is $\mathsf{Count}_{\leq}(\lfloor Dt/s \rfloor, d, m)$ (this is the number of multivariate Hasse derivatives).

  It remains to address the runtime of Corollary 5.1, which is bounded above by $\left|\mathsf{HW}_{\min(\lfloor Dt/s \rfloor,m),d,m}\right| \cdot \mathsf{poly}(m, d, \log q)$. This is identical to the case of server runtime per query, so assuming $\boxed{\theta t/s \leq d/(d+1)}$ this again simplifies to:

$$2^{(H(\theta t/s)+\theta t \log d/s+o(1))m} = n^{\frac{s \cdot H(\theta t/s)+\theta t \log d}{s \cdot \mathsf{CntExp}_{\leq}(\theta,d)}+o(1)}.$$ □

Special cases of Theorem 5.3. Setting the collusion threshold $t = 1$ and the individual degree $d = 1$ in Theorem 5.3, we obtain a scheme that strictly improves on state-of-the-art PIR, when the number of servers $s > 2$ is constant: it achieves the same server time per query as prior work [GLM+25, Theorem 6.4], with storage that is polynomially smaller by a factor of $n^{H(\theta/s)/H(\theta)+o(1)}$. In particular:

Corollary D.10 (Information-theoretic, one-private PIR with preprocessing)**.** For any number of servers $s \geq 2$, database size $n \in \mathbb{N}$, prime $q \geq s$, and constant $0 < \theta \leq 1/2$, there exists an information-theoretically one-private PIR-with-preprocessing scheme with s servers, where: the server storage and preprocessing time are $n^{\log q/H(\theta)+o(1)}$, the server time per query is $n^{H(\theta/s)/H(\theta)+o(1)}$, and the communication and client time per query are $n^{H(\theta/s)/H(\theta)+o(1)}$.

Proof Sketch. Note that this is a valid setting of parameters since $\theta t/s \le d/(d+1) \Leftrightarrow \theta \le s/2$, which is ensured for any $\theta \le 1/2$. □

As a second design point, we observe that setting the individual degree d to be as large as possible minimizes the scheme's storage, which becomes near linear in the database size n:

Corollary D.11 (Information-theoretic, one-private PIR with preprocessing with near-linear storage)**.** For any number of servers $s \ge 2$, database size $n \in \mathbb{N}$ and prime $s \le q \le 2s$, there is an information-theoretically 1-private PIR-with-preprocessing scheme with $s \ge 2$ servers, where: the server storage is $O(n \log n)$, the server preprocessing time is $O(n(\log n)^3)$, and the server time per query, client time per query, and communication are all $n^{\alpha+o(1)}$ for $\alpha = (H\left(\frac{q-1}{2s}\right) + \frac{q-1}{2s} \cdot \log(q-1))/\log q$.
If s is prime and $q = s$, the exponent α in the server time simplifies to $1 + \frac{1}{\log s} - \left(\frac{s+1}{2s}\right)\frac{\log(s+1)}{\log s}$, which approaches $1/2$ as s grows.

Proof. The proof sets $d = q-1$ and $\theta = d/2$ in Theorem 5.3. Note that this is valid since $\theta t/s \le d/(d+1) \Leftrightarrow d/(2s) \le d/(d+1) \Leftrightarrow q \le 2s$. When $\theta = d/2$, per Theorem A.2, we have $q_\theta(x) = \sum_{i=0}^{d}(d/2 - i) \cdot x^i$. By inspection, this is zero at $x = 1$, giving $p(\theta) = 1$. Therefore, we have

$$\mathsf{CntExp}_{\le}(\theta, d) = \mathsf{CntExp}(\theta, d) = \log(d+1) - \frac{d}{2}\log 1 = \log(d+1) = \log q.$$

The exponent in the server time is thus:

$$\frac{s \cdot H(\theta/s) + \theta t \log d}{s \cdot \mathsf{CntExp}_{\le}(\theta, d)} = \frac{H\left(\frac{q-1}{2s}\right) + \frac{q-1}{2s} \cdot \log(q-1)}{\log q}.$$

To see the server storage, note that with $\theta = d/2$ we have $n = \mathsf{Count}_{\le}((q-1)m/2, q-1, m) \in [q^m/2, q^m]$[8], and the server storage is $O(q^m D) = O(nD) = O(n \log n)$.

To see the preprocessing time, we need time $O(n(\log n)^3)$ for Theorem B.3 and time $O(q^m \cdot m \cdot D) = O(n(\log n)^2)$ for Lemma D.6. □

D.3 Additional Material on Construction Optimizing Server Time

Proof of Theorem 5.4.

Proof. Recall that $\theta t/s < d/2$ implies that $\mathsf{CntExp}_{\le}(\theta t/s, d) = \mathsf{CntExp}(\theta t/s, d)$. To prove the theorem statement, we make the following changes to the construction in Fig. 5:

1. In the preprocessing phase, the servers store a truth table of each of the homogeneous slices of the database polynomial $f_0, f_1, \ldots, f_D$, and of the Hasse derivatives of each slice up to order $\lfloor Dt/s \rfloor$:

   $$\mathsf{Table} = \left\{ \overline{\partial}_{\mathbf{e}} f_\ell(\mathbf{x}) \,:\, \mathbf{x} \in \mathbb{F}_q^m, \ell \in \{0, 1, \ldots, D\}, \mathbf{e} \in S_{\le \lfloor \ell t/s \rfloor, d, m} \right\}.$$

   By Lemma D.6, the storage and preprocessing time are now

   $$q^m \cdot \mathsf{Count}_{\le}(Dt/s, d, m) \cdot 2^{o(m)} = n^{\frac{\log q + \mathsf{CntExp}(\theta t/s, d)}{\mathsf{CntExp}_{\le}(\theta, d)} + o(1)}.$$

[8] This can be seen with a bijective argument; the number of tuples in $[0, q-1]^m$ with sum $< (q-1)m/2$ is equal to the number with sum $> (q-1)m/2$ by replacing $(x_1, \ldots, x_m)$ with $(q-1-x_1, \ldots, q-1-x_m)$.

2. The querying phase remains unchanged. The user sends a query point $\mathsf{qu}_j \in \mathbb{F}_q^m$ to server j.

3. In the answering phase, server j responds with the evaluations of the polynomials $f_0, f_1, \ldots, f_D$ and of all of their Hasse derivatives up to order $\lfloor Dt/s \rfloor$, at the received query point qu_j:

$$\mathsf{ans}_j := \left\{ \overline{\partial}_{\mathbf{e}} f_\ell(\mathsf{qu}_j) \ : \ \ell \in \{0, 1, \ldots D\}, \mathbf{e} \in S_{\leq \lfloor \ell t/s \rfloor, d, m}. \right\}.$$

   The server retrieves each of these values with a single table lookup.

4. In the reconstruction phase, the user directly applies the chain rule (skipping the finite-differences technique). The runtime of this step is

$$\mathsf{Count}_{\leq}(Dt/s, d, m) \cdot \mathsf{poly}(Dt/s, t, \log q) = n^{\frac{\mathsf{CntExp}_{\leq}(\theta t/s, d)}{\mathsf{CntExp}(\theta, d)} + o(1)}.$$

   By Lemma D.4, Hermite interpolation runs in time $\mathsf{poly}(D, t, \log q) = n^{o(1)}$. □

E Additional Material on Evaluation

Implementation of PIR with preprocessing. The Answer routine in our protocol makes many lookups to a large table; a handful of optimizations in our code are critical to achieving good performance.

First, we select appropriate data types: we store the preprocessed Table as a densely-packed array of bytes. We represent each evaluation point in $\mathbb{F}_2^m$ as a 64-bit integer, which corresponds to the index in the array that holds the associated polynomial evaluation. (In our experiments, it always holds that $m \leq 64$, since the lookup table has size 2^m bits.) Second, for databases with records of length $\ell \geq 1$ bits, we interleave the data in the array in a way that, on each memory access, the servers read ℓ consecutive bits from the array. This amortizes the cost of the random-access lookups as the record-length ℓ grows large. Finally, in the Answer routine, we iterate through the points in the Hamming ball (i.e., the points $\mathsf{qu} + \mathbf{e} \in \mathbb{F}_2^m$ for $\|\mathbf{e}\|_1 \leq \lfloor D/2 \rfloor$, on query point $\mathsf{qu} \in \mathbb{F}_2^m$) in lexicographic order, sorted by the vector $\mathbf{e}$. This ensures that nearby memory accesses often occur in sequence, taking advantage of locality and caching.

Baseline: XOR PIR. In this scheme, each server represents the n-bit database as a matrix $\mathbf{D} \in \mathbb{F}_2^{\sqrt{n} \times \sqrt{n}}$. The user sends each server a query $\mathbf{x} \in \mathbb{F}_2^{\sqrt{n}}$; each server responds with $\mathbf{Dx}$, i.e., the XOR of all columns of this matrix corresponding to a "1" in the query vector. This requires a linear scan over the database and is bottlenecked by the machine's memory bandwidth (which governs how fast the database can be read from RAM). XOR PIR is information-theoretically private.