PIR through the Lens of Probabilistic Inference

Dana Dachman-Soled[a]

University of Maryland, USA

Abstract. Private Information Retrieval (PIR) allows a client to retrieve an entry of a database while revealing no information about the requested index to the servers. Classic information-theoretic PIR protocols of Chor et al. (FOCS '95, JACM '98) and Ambainis (ICALP '97) achieve sublinear client computation and communication complexity with a constant number of servers, but require each server to perform computation linear in the database size. In this work, we ask for which classes of structured databases the server computation can be substantially reduced while preserving the communication complexity and privacy guarantees of these protocols.
Our main contribution is a new connection between server computation in information-theoretic PIR and probabilistic inference. We show that the computation performed by every server in both the Chor et al. protocol and the recursive protocol of Ambainis can be viewed as computing a marginal probability of a product distribution. Consequently, efficient exact probabilistic inference algorithm immediately yields a faster server algorithm for PIR. We formalize this connection through the notion of *inference complexity*, which captures the complexity of evaluating these marginals.
Using this framework, we obtain improved server computation for several broad classes of functions, including read-once arithmetic formulas, graphical models of bounded treewidth (and consequently arithmetic circuits of bounded treewidth), functions with low communication complexity, and functions with sparse Fourier representations. The resulting protocols preserve the communication complexity, client computation, and number of servers of the underlying PIR constructions while substantially reducing server computation for these structured function classes.
Finally, we complement our algorithms with conditional lower bounds. Assuming the randomized Circuit Strong Exponential Time Hypothesis (rCSETH), we show that, for every constant $\varepsilon > 0$, there exists a constant c such that no uniform algorithm can compute the first (virtual) server's response in the Chor et al. or Ambainis protocol on all n-input Boolean circuits of size at most n^c in time $O(N^{1-\varepsilon})$, where $N = 2^n$. Thus, under rCSETH, substantially sublinear server computation cannot be achieved uniformly for all polynomial-size Boolean circuits.

Keywords: Private information retrieval · Information-theoretic security · Server computation · Probabilistic inference · Treewidth · Fine-grained complexity

1 Introduction

Private Information Retrieval (PIR) allows a client to retrieve an entry from a database while keeping the index of the requested entry private from the servers holding copies of the database. In an information-theoretic PIR protocol, this privacy guarantee holds even against computationally unbounded servers. The main goal in PIR is to minimize the communication between the client and the servers, while also keeping the number of servers and the computational requirements of the client and servers small. Since the seminal work of Chor et al. [CGKS95, CGKS98], a large body of research has studied

E-mail: danadach@umd.edu (Dana Dachman-Soled)
[a]Supported in part by NSF grant CCF-2552210.

information-theoretic PIR, leading to protocols with increasingly efficient communication complexity. In particular, the protocols of Chor et al. [CGKS95] and Ambainis [Amb97] achieve communication complexity that is sublinear in the database size N while using a constant number of servers. However, these protocols require each server to perform computation that is linear in the size of the database. Furthermore, it is not clear whether the server computation can be reduced even when the database is highly structured, namely when it corresponds to the truth table of a function f, where the entry indexed by $[i_1, \ldots, i_d] \in [N^{1/d}]^d$ contains the value $f(i_1, \ldots, i_d)$. Throughout the paper, we assume d is a constant and treat it as such in the asymptotic notation.

Motivating Application: Privacy-Preserving Access Control. Consider a setting in which a collection of non-colluding servers holds an access-control policy and a decryption key for an encrypted file. A client wishes to obtain the decryption key if its credentials satisfy the policy, without revealing its credentials or whether they satisfy the policy to any individual server. This can be modeled as PIR over a database representing a function f that maps each possible credential tuple to the decryption key if the policy is satisfied, and to zero otherwise. Since the client does not know the decryption key, it cannot simply evaluate the function locally. Moreover, in the semi-honest setting, privacy against the client can be achieved by having the servers compute the XOR of their PIR responses through a round-robin protocol, masked by a random value chosen by the client, so that the client learns only the final output rather than the individual server responses. This aggregation preserves the asymptotic communication complexity of the underlying PIR protocol. When the access-control policy admits a compact representation, such as a read-once formula or a bounded-treewidth circuit, our techniques substantially reduce the server computation required to implement this functionality via the Chor et al. [CGKS95] or Ambainis [Amb97] protocols.

Our contributions. In this work, we revisit the PIR protocols of Chor et al. [CGKS95] and Ambainis [Amb97] and show that, when the database corresponds to the truth table of a function f, the complexity of the server computation is governed by a new measure that we call the *inference complexity* of f. This characterization allows us to identify broad classes of functions for which the server computation can be significantly reduced while preserving the communication complexity of the original protocols. Specifically, for a database of size N, the basic protocol of Chor et al. (reviewed in Section 3) has communication complexity $O(N^{1/d})$, client computation $O(N^{1/d})$, and uses 2^d servers, but requires each server to perform $\Omega(N)$ computation. The protocol of Ambainis (reviewed in Section 6.1) achieves the same communication complexity and client computation while using only $\frac{d+1}{2}$ servers. It likewise requires $\Omega(N)$ computation per server at the top level of the recursion, and $\Omega\big(N^{1-2(\ell-1)/d}\big)$ computation at the ℓ-th level of the recursion.

Our main result is that the same characterization applies to both the basic Chor et al. protocol and the recursive Ambainis protocol. The key connection we establish is that functions with low *inference complexity* admit more efficient server algorithms. We elaborate in the next subsection on our notion of *inference complexity* and explain why it leads to faster server computation.

Using this connection, together with several additional techniques, we obtain faster server algorithms for several broad classes of functions, including read-once arithmetic formulas, functions representable as graphical models of bounded treewidth (and, in particular, arithmetic circuits of bounded treewidth), functions with low communication complexity, and functions with sparse Fourier representations. The resulting server complexities are summarized in Table 1.

We also complement our algorithmic results with lower bounds. Under the Randomized Circuit Strong Exponential Time Hypothesis (rCSETH, see Definition 5)–a Circuit-SAT variant of SETH–we show that computing the output of the first (virtual) server in the

Table 1: Server computation for the function classes considered in this paper. Here $N^{1/d}$ is the domain size of each variable. The running times apply to both the protocol of Chor et al. [CGKS95] and the protocol of Ambainis [Amb97]. The Chor et al. protocol uses 2^d servers, while the Ambainis protocol achieves the same communication complexity using $(d+1)/2$ servers.

Function class	**Parameter(s)**	**Server computation**
Read-once arithmetic formulas	formula size s	$O(s\,N^{1/d}\log N)$
Graphical models	treewidth w, s variables	$O(s\,(N^{1/d})^{w+1})$
Communication complexity	communication cost ρ	$O(2^\rho N^{1/d})$
Sparse Fourier representation	Fourier sparsity s	$O(s\,N^{1/d})$

basic Chor et al. or Ambainis protocols requires time $\Omega(N^{1-\varepsilon})$, for any constant ε, for a family of polynomial-size Boolean circuits.

Recent work has announced subexponential-time algorithms for fixed-width k-SAT, challenging ETH and SETH [Ope26, Ant26]. However, these results do not immediately refute the randomized Circuit Strong Exponential Time Hypothesis (rCSETH) used in our lower bounds. In particular, converting an arbitrary Boolean circuit with n inputs and polynomially many gates into a 3-CNF formula may introduce polynomially many auxiliary variables. Consequently, the new SAT algorithms do not directly yield a $2^{(1-\varepsilon)n}$-time algorithm for satisfiability of arbitrary polynomial-size Boolean circuits. Thus, rCSETH remains a potentially viable hardness assumption despite these developments.

1.1 Inference complexity and fast server computation

The key observation underlying our work is that the computation performed by the server in the Chor et al. protocol is naturally an instance of probabilistic inference.

To see this, let the database be viewed as the truth table of a function $f : [N^{1/d}]^d \to \{0,1\}$. For a fixed query, the subsets chosen by the client induce indicator vectors $\mathbf{M}^1, \ldots, \mathbf{M}^d \in \{0,1\}^{N^{1/d}}$, one for each coordinate. The server's response can then be written as

$$\sum_{x \in [N^{1/d}]^d} f(x) \prod_{i=1}^{d} \mathbf{M}^i[x_i] \pmod 2.$$

Thus, the server is not evaluating f at a single point, but rather computing a weighted sum of f over a Cartesian product of subsets.

Normalizing the vectors, $\mathbf{V}^i = \frac{\mathbf{M}^i}{\|\mathbf{M}^i\|_1}$, transforms them into probability distributions, where we denote by $X_i \sim \mathbf{V}^i$ sampling X_i from the distribution that sets $X_i = x$ with probability $\mathbf{V}^i[x]$. Consequently,

$$\sum_{x} f(x) \prod_{i=1}^{d} \mathbf{V}^i[x_i] = \Pr_{X_i \sim \mathbf{V}^i}[f(X) = 1],$$

where the random variables $X_1, \ldots, X_d$ are sampled independently. Up to the known scaling factor $\prod_i \|\mathbf{M}^i\|_1$, the Server's computation is therefore exactly the problem of computing the marginal probability that f evaluates to 1 under a product distribution.

This perspective motivates the notion of the *inference complexity* of a function, which informally measures the complexity of computing such marginals for arbitrary product distributions. A low inference complexity immediately translates into a faster Server algorithm, since the Server computation is precisely such an inference problem.

A priori, this interpretation applies only to the basic protocol of Chor et al. One of our main technical contributions is showing that the same viewpoint extends to the recursive protocol of Ambainis. We prove that every recursive query still induces a product distribution with respect to the original variables. Consequently, the Server computation at every level of the recursion can again be expressed as a probabilistic inference problem, allowing exactly the same inference algorithms to be applied.

This connection enables us to leverage algorithmic techniques from a variety of areas. In particular, efficient inference algorithms for graphical models yield faster Server computation for functions of bounded treewidth; algorithms for read-once arithmetic formulas lead to efficient PIR for such formulas; communication protocols give rise to efficient inference procedures for functions of low communication complexity; and sparse Fourier representations similarly lead to faster Server algorithms. Thus, the notion of inference complexity provides a unifying explanation for why these seemingly unrelated classes of functions admit improved Server computation.

1.2 Related Work

Information-Theoretic PIR. Information-theoretic private information retrieval (PIR) was introduced by Chor, Goldreich, Kushilevitz, and Sudan [CGKS95], who gave the first multi-server protocols achieving sublinear communication. Their construction uses 2^d non-communicating servers and achieves communication complexity $O(N^{1/d})$. Ambainis [Amb97] subsequently introduced a recursive construction that reduces the number of servers to $\frac{d+1}{2}$ while preserving sublinear communication, obtaining communication complexity $O(N^{1/d})$. A long line of subsequent work focused primarily on further reducing the communication complexity of information-theoretic PIR, leading to protocols with nearly optimal communication while using a constant number of servers, including the works of Beimel, Ishai, Kushilevitz, and Raymond [BIKR02], Yekhanin [Yek07], Efremenko [Efr09], Dvir and Gopi [DG15], and Ghasemi, Kopparty, and Sudan [GKS25]. These works optimize the communication cost and, in some cases, the number of servers, but they generally continue to require the servers to process a linear amount of data.

Our work is orthogonal to this line of research. Rather than improving communication complexity, we ask when the *server computation* can be reduced while preserving the communication complexity of the underlying PIR protocol. We show that for the protocols of Chor et al. [CGKS95] and Ambainis [Amb97], the server computation is governed by the inference complexity of the function represented by the database. This perspective yields faster server algorithms for several natural classes of functions without changing the communication complexity of the underlying PIR protocol.

Server computation in PIR. The computational cost incurred by the servers has long been recognized as one of the principal bottlenecks in PIR systems. In the original information-theoretic protocols of Chor et al. [CGKS95] and Ambainis [Amb97], each server performs a linear scan over the database (or the recursively constructed auxiliary databases), resulting in server computation proportional to the database size. Subsequent work on computational PIR has similarly devoted considerable attention to reducing server-side computation, often through the use of homomorphic encryption, preprocessing, batching, or specialized data structures [KO97, GR05, AMBFK16, ACLS18, ACL+21, BCG+23]. Our work is complementary to this line of research. Rather than changing the PIR protocol itself, we identify broad classes of database functions for which the server computation in the information-theoretic protocols of Chor et al. and Ambainis can be asymptotically reduced while preserving the original communication complexity.

Probabilistic inference. Probabilistic inference is a fundamental computational problem that has been extensively studied in machine learning, artificial intelligence, and

statistics [Pea88, KF09, Dar11]. Exact inference is generally intractable, but efficient algorithms are known for broad classes of structured probabilistic models, including graphical models of bounded treewidth, arithmetic circuits, and related representations [LS88, Dar11, KF09, PCdG+19]. Common exact inference algorithms include variable elimination, belief propagation on trees, junction-tree algorithms, and arithmetic-circuit evaluation [Pea88, LS88, Dar11, PCdG+19]. These algorithms exploit structural properties of the underlying model to reduce computation while producing exact marginal probabilities. Our work establishes a connection between this literature and information-theoretic PIR. Specifically, we show that the server computation in the protocols of Chor et al. [CGKS95] and Ambainis [Amb97] can be viewed as exact probabilistic inference over a product distribution. Consequently, any exact inference algorithm immediately yields a corresponding algorithm for reducing the server computation of these PIR protocols. This perspective allows us to leverage a broad range of existing inference algorithms to obtain improved server complexities for several natural classes of functions.

2 Notation and Preliminaries

For a natural number $n \in \mathbb{N}$, we denote by $[n]$ the set of integers $\{0, \ldots, n-1\}$ and we denote by $[n]_+$ the set of integers $\{1, \ldots, n\}$. Throughout this paper, uppercase letters denote random variables, while lowercase letters represent their realizations.

For a set S we denote by $|S|$ the size of the set. For a vector $\mathbf{v}$ we denote by $\|\mathbf{v}\|_1$ its ℓ_1 norm.

For a nonempty subset $S \subseteq [N^{1/d}]$, let $\mathbf{M}_S$ denote the indicator vector of S, and let $\mathbf{V}_S := \frac{\mathbf{M}_S}{\|\mathbf{M}_S\|_1}$ denote the corresponding probability distribution. In particular, if $X \sim \mathbf{V}_S$, then for $x \in [N^{1/d}]$, $\Pr[X = x] = \mathbf{V}_S[x]$.

3 The Basic PIR Scheme of Chor et al. [CGKS95]

Let $f : [N^{1/d}]^d \to \{0,1\}$ denote the database, and let $(x_1^*, \ldots, x_d^*) \in [N^{1/d}]^d$ be the index that the client wishes to retrieve. For a set $S \subseteq [N^{1/d}]$ and an element $t \in [N^{1/d}]$, define

$$S \oplus t := \begin{cases} S \cup \{t\}, & \text{if } t \notin S, \\ S \setminus \{t\}, & \text{if } t \in S. \end{cases}$$

Equivalently, $S \oplus t$ is the symmetric difference of S and the singleton set $\{t\}$.

Client. For each coordinate $j \in [d]_+$, the client samples a random subset $S_j^0 \subseteq [N^{1/d}]$ by including each element independently with probability $1/2$, and defines

$$S_j^1 = S_j^0 \oplus x_j^*.$$

The client sends $(S_1^{id_1}, \ldots, S_d^{id_d})$ to the server whose identity is $id = (id_1, \ldots, id_d) \in \{0,1\}^d$.

Server. Each server computes: $\mathsf{out}_{id} = \bigoplus_{(x_1,\ldots,x_d) \in S_1^{id_1} \times \cdots \times S_d^{id_d}} f(x_1, \ldots, x_d)$, and returns out_{id} to the client.

Reconstruction. The client outputs: $\mathsf{out} = \bigoplus_{id \in \{0,1\}^d} \mathsf{out}_{id}$.

4 An Inference-Complexity View of [CGKS95]

The protocol of Section 3 can be viewed from a probabilistic perspective. Instead of explicitly summing the values of f over a Cartesian product of subsets, we view each subset as defining a probability distribution. If any of the subsets are empty, the server simply returns 0. Henceforth, we therefore assume WLOG that the subsets are non-empty.

Client. The client performs exactly the same computation as in the original protocol. For each $j \in [d]_+$, it samples a random subset $S_j^0 \subseteq [N^{1/d}]$, sets

$$S_j^1 = S_j^0 \oplus x_j^*,$$

and sends $(S_1^{id_1}, \ldots, S_d^{id_d})$ to the server with identity $id = (id_1, \ldots, id_d) \in \{0,1\}^d$.

Server. Instead of explicitly evaluating f on every point of $S_1^{id_1} \times \cdots \times S_d^{id_d}$, the server computes

$$P = \Pr_{(X_1,\ldots,X_d)\sim \mathbf{V}_{S_1^{id_1}} \times \cdots \times \mathbf{V}_{S_d^{id_d}}} [f(X_1, \ldots, X_d) = 1].$$

The server returns

$$\mathsf{out}_{id} = \left(P \cdot \prod_{j=1}^{d} |S_j^{id_j}| \right) \bmod 2.$$

Reconstruction. The client outputs $\mathsf{out} = \bigoplus_{id \in \{0,1\}^d} \mathsf{out}_{id}$.

The above protocol is identical to the protocol of Chor et al. Indeed,

$$P = \frac{\left|\{x \in S_1^{id_1} \times \cdots \times S_d^{id_d} : f(x) = 1\}\right|}{\prod_{j=1}^{d} |S_j^{id_j}|},$$

and therefore

$$\begin{aligned} \mathsf{out}_{id} &= \left(P \prod_{j=1}^{d} |S_j^{id_j}| \right) \bmod 2 \\ &= \left|\{x \in S_1^{id_1} \times \cdots \times S_d^{id_d} : f(x) = 1\}\right| \bmod 2 \\ &= \bigoplus_{x \in S_1^{id_1} \times \cdots \times S_d^{id_d}} f(x), \end{aligned}$$

which is exactly the value returned by the original protocol. Consequently, correctness and privacy follow immediately from the analysis of the original protocol.

This reformulation reveals a direct connection between server computation in PIR and exact probabilistic inference and motivates the following measure of the complexity of exactly evaluating such probabilities.

Definition 1 (Inference Complexity)**.** Let $f : [N^{1/d}]^d \to \{0,1\}$ be a function, and let $\mathcal{S}$ denote the collection of all Cartesian products

$$S = S_1 \times \cdots \times S_d, \qquad S_i \subseteq [N^{1/d}].$$

For non-empty $S \in \mathcal{S}$, the *inference complexity* of f with respect to S, denoted by $\mathsf{Inf}_S(f)$, is the running time required to compute

$$P_S = \Pr_{(X_1,\ldots,X_d)\sim \mathbf{V}_{S_1}\times\cdots\times\mathbf{V}_{S_d}} [f(X_1,\ldots,X_d) = 1] .$$

The *inference complexity* of f is $\mathsf{Inf}(f) = \max_{S\in\mathcal{S}\setminus\emptyset} \mathsf{Inf}_S(f)$.

The following proposition provides the formal connection between inference complexity and server runtime.

Proposition 1. *Each server in the PIR protocol can compute its output in time $O(\mathsf{Inf}_S(f))$, up to the cost of the final arithmetic operation required to recover the PIR answer.*

Thus, every exact inference algorithm for the resulting product distribution immediately gives a PIR server algorithm with the corresponding running time. In Section 5, we instantiate this connection for several classes of structured functions, obtaining substantially faster server algorithms while preserving the communication complexity and privacy guarantees of the underlying PIR protocol.

5 Instantiations of the Framework

5.1 Read-Once Arithmetic Formulas

As our first application of the probabilistic reformulation, we consider Boolean functions that can be represented by read-once arithmetic formulas.

Definition 2 (Read-Once Arithmetic Formula)**.** A *read-once arithmetic formula* over a field $\mathbb{F}$ is a rooted tree in which each leaf is labeled by either a variable x_i or a field element $a \in \mathbb{F}$, and each internal node is labeled by one of the operations $+, \quad \times, \quad -, \quad (\cdot)^{-1}$. The operations $+$ and $\times$ are binary, while $-$ and $(\cdot)^{-1}$ are unary. Each variable x_i occurs as a leaf at most once in the formula.

The value computed by the formula is defined recursively in the natural way. A leaf labeled by x_i evaluates to x_i, and a leaf labeled by $a \in \mathbb{F}$ evaluates to a. An internal node labeled by $+$ or $\times$ evaluates to the sum or product, respectively, of the values computed by its two children. An internal node labeled by $-$ evaluates to the additive inverse of the value computed by its child, and an internal node labeled by $(\cdot)^{-1}$ evaluates to the multiplicative inverse of the value computed by its child, with the convention that $0^{-1} = 0$.

The key observation is that the probability distribution of the output of each gate can be computed solely from the probability distributions of its inputs. Consequently, the formula can be evaluated in a bottom-up fashion, maintaining at each gate the probability mass function (PMF) of its output. Using FFT-based convolution algorithms, each gate can be processed in time $O\big(N^{1/d}\log N\big)$, yielding an overall running time of $O\big(sN^{1/d}\log N\big)$ for a formula of size s. Details follow.

5.1.1 A Warmup

As a warmup we prove the following lemma with higher time-complexity. In the next section we show how to improve the time complexity to obtain our main result.

Lemma 1. *Let f be represented by a read-once arithmetic formula of size s. Then the server's computation in the PIR protocol can be performed in $O\big(sN^{2/d}\big)$ time.*

Proof. Suppose f is represented by a read-once arithmetic formula of size s over $\mathbb{Z}_q$, where $q = N^{1/d}$ is prime. Every internal gate is either addition or multiplication modulo q.

For every gate G, we associate a vector $\mathbf{V}_G \in \mathbb{R}^q$, where $\mathbf{V}_G[x] = \Pr[G = x]$. Thus, $\mathbf{V}_G$ is the probability mass function of the random variable computed at gate G under the product distribution induced by the client. The PMFs are computed in a bottom-up traversal of the formula.

Addition gate. Suppose an addition gate has input PMFs $\mathbf{V}_L$ and $\mathbf{V}_R$. The output distribution is the convolution

$$\mathbf{V}_{\text{out}}[z] = \sum_{x \in \mathbb{Z}_q} \mathbf{V}_L[x]\, \mathbf{V}_R[z - x].$$

A direct implementation requires $O(q^2) = O(N^{2/d})$ time.

Multiplication gate. Suppose a multiplication gate has input PMFs $\mathbf{V}_L$ and $\mathbf{V}_R$.

The probability that the output is zero is

$$\mathbf{V}_{\text{out}}[0] = \mathbf{V}_L[0] + \mathbf{V}_R[0] - \mathbf{V}_L[0]\mathbf{V}_R[0].$$

For every nonzero $z \in \mathbb{Z}_q^*$,

$$\mathbf{V}_{\text{out}}[z] = \sum_{x \in \mathbb{Z}_q^*} \mathbf{V}_L[x]\, \mathbf{V}_R[zx^{-1}].$$

Again, this computation requires $O(q^2)$ time.

Additive and Multiplicative Inverse gates. Suppose an additive inverse gate has input PMF $\mathbf{V}_{\text{in}}$. The output distribution is $\mathbf{V}_{\text{out}}[z] = \mathbf{V}_{\text{in}}[-z]$. A direct implementation requires $O(q) = O(N^{1/d})$ time.

Suppose a multiplicative inverse gate has input PMF $\mathbf{V}_{\text{in}}$ The output distribution is $\mathbf{V}_{\text{out}}[z] = \mathbf{V}_{\text{in}}[z^{-1}]$, for $z \neq 0$ and $\mathbf{V}_{\text{out}}[0] = \mathbf{V}_{\text{in}}[0]$. A direct implementation requires $O(q) = O(N^{1/d})$ time.

Output gate. The output gate maps $\mathbb{Z}_q$ to $\{0, 1\}$ by outputting the t-th bit of the representation of a field element z. Suppose the output gate has input PMF $\mathbf{V}_{\text{in}}$. For $b \in \{0, 1\}$, the output distribution is $\mathbf{V}_{\text{out}}[b] = \sum_{z : z[t]=b} \mathbf{V}_{\text{in}}[z]$. A direct implementation requires $O(q) = O(N^{1/d})$ time.

The total time for a formula of size s is therefore $O(s \cdot q^2) = O(s \cdot N^{2/d})$. □

5.1.2 FFT Acceleration

We now show how to improve the time complexity of the algorithm given above and we prove the following theorem:

Theorem 1. *Let $q = N^{1/d}$ be prime, and let f be represented by a read-once arithmetic formula of size s over $\mathbb{Z}_q$. Then the server's computation in the PIR protocol can be performed using $O\big(sN^{1/d} \log N\big)$ arithmetic operations on $O(\log N)$-bit precision numbers.*

Proof. The quadratic-time convolutions from Section 5.1.1 can be replaced by FFT-based convolutions.

For addition gates, the convolution theorem gives

$$\mathbf{V}_{\text{out}} = \mathsf{DFT}_q^{-1}(\mathsf{DFT}_q(\mathbf{V}_L) \odot \mathsf{DFT}_q(\mathbf{V}_R)),$$

where $\odot$ denotes pointwise multiplication.

For multiplication gates, we exploit that $\mathbb{Z}_q^*$ is cyclic. Let g be a generator of $\mathbb{Z}_q^*$, and define

$$\log \mathbf{V}_L[i] = \mathbf{V}_L[g^i], \qquad \log \mathbf{V}_R[i] = \mathbf{V}_R[g^i].$$

The convolution becomes an ordinary cyclic convolution over $\mathbb{Z}_{q-1}$:

$$\log \mathbf{V}_{\text{out}} = \mathsf{DFT}_{q-1}^{-1}(\mathsf{DFT}_{q-1}(\log \mathbf{V}_L) \odot \mathsf{DFT}_{q-1}(\log \mathbf{V}_R)).$$

The values are then mapped back via

$$\mathbf{V}_{\text{out}}[g^i] = \log \mathbf{V}_{\text{out}}[i],$$

while

$$\mathbf{V}_{\text{out}}[0] = \mathbf{V}_L[0] + \mathbf{V}_R[0] - \mathbf{V}_L[0]\mathbf{V}_R[0].$$

To ensure exactness, observe that for every gate G, the probabilities in its output distribution are integer multiples of $1/D_G$, where $D_G = \prod_{i \in I_G} |S_i| \leq N$, and I_G is the set of variables appearing in the subformula rooted at G. Thus, it suffices to compute each convolution with entrywise additive error strictly less than $1/(2N)$, and then round each output probability to the nearest multiple of $1/D_G$. Standard FFT algorithms with $O(\log N)$ bits of precision achieve the required accuracy, with $O(q \log q)$ arithmetic operations per gate. Since rounding recovers the exact probability distribution after every gate, numerical errors do not accumulate across the formula. Thus, every gate can be evaluated using $O\big(N^{1/d} \log N\big)$ arithmetic operations on $O(\log N)$-bit precision numbers, implying the theorem. □

5.2 Graphical Models of Bounded Treewidth

The following exposition closely follows that of [PCdG+19]. A probabilistic graphical model consists of a collection of random variables $X = (X_1, \ldots, X_s)$, where each variable X_i takes values in a finite domain Λ_i. The joint state space is $\Lambda = \prod_{i=1}^{s} \Lambda_i$. For any subset $A \subseteq [s]$, we write X_A, x_A, and Λ_A for the corresponding variables, assignments, and domains.

A graphical model is specified by a collection of factors $\{\psi_B\}_{B \in \mathcal{B}}$, where each factor $\psi_B : \Lambda_B \to \mathbb{R}_{\geq 0}$ depends only on the variables indexed by B. The resulting factorization of the joint probability function is

$$p(x) = \frac{1}{Z} \prod_{B \in \mathcal{B}} \psi_B(x_B), \quad \text{where} \quad Z = \sum_{x \in \Lambda} \prod_{B \in \mathcal{B}} \psi_B(x_B)$$

is the partition function. In our applications, the variables are partitioned into three groups: $X = (X^{\mathsf{in}}, X^{\text{out}}, X^{\text{lat}})$, corresponding to the input variables, output variables, and latent variables of the computation. Given a setting of $(X^{\mathsf{in}} = x^{\mathsf{in}}, X^{\text{out}} = x^{\text{out}}, X^{\text{lat}} = x^{\text{lat}})$, the *unnormalized* graphical model $g(x) = \prod_{B \in \mathcal{B}} \psi_B(x_B)$ will be set to 1 if $x = (x^{\mathsf{in}}, x^{\text{out}}, x^{\text{lat}})$ is *consistent* and $g(x)$ will be set to 0 if $x = (x^{\mathsf{in}}, x^{\text{out}}, x^{\text{lat}})$ is *inconsistent*.

To incorporate the client's PIR query, we introduce one unary factor for each input coordinate. If the client sent non-empty subset $S_j^{id_j}$, We define

$$\mathbf{M}_{S_j^{id_j}}(x_j^{\mathsf{in}}) = \mathbf{M}_{S_j^{id_j}}[x_j^{\mathsf{in}}] = \begin{cases} 1 & x \in S_j^{id_j}, \\ 0 & \text{otherwise.} \end{cases}$$

$$\mathbf{V}_{S_j^{id_j}}(x_j^{\mathsf{in}}) = \mathbf{V}_{S_j^{id_j}}[x_j^{\mathsf{in}}] = \begin{cases} \frac{1}{\|\mathbf{M}_{S_j^{id_j}}\|_1} & x \in S_j^{id_j}, \\ 0 & \text{otherwise.} \end{cases}$$

We now state the main theorem of this section.

Theorem 2. *Assume the function $f(x^{\mathsf{in}})$ is associated with an unnormalized graphical model:*

$$g(x) = \prod_{B \in \mathcal{B}} \psi_B(x_B),$$

such that for all $x = x^{\mathsf{in}} || x^{\mathsf{out}} || x^{\mathsf{lat}}$,

- *If $f(x^{\mathsf{in}}) = x^{\mathsf{out}}$ then there exists a unique setting of x^{lat} such that $g(x) = 1$.*
- *If $f(x^{\mathsf{in}}) \neq x^{\mathsf{out}}$ then for all settings of x^{lat}, $g(x) = 0$.*

Let

$$\Phi(x) = \prod_{B \in \mathcal{B}} \psi_B(x_B) \prod_{j=1}^{d} \mathbf{V}_{S_j^{id_j}}(x_j^{\mathsf{in}}) = \frac{1}{Z} \prod_{B \in \mathcal{B}} \psi_B(x_B) \prod_{j=1}^{d} \mathbf{M}_{S_j^{id_j}}(x_j^{\mathsf{in}}),$$

where $Z := \prod_{j=1}^{d} \|\mathbf{M}_{S_j^{id_j}}\|_1$.

Then the server's computation in the PIR protocol can be performed by computing

$$P = \Pr[X^{\text{out}} = 1] = \sum_{x^{\mathsf{in}}, x^{\text{lat}}} \Phi(x^{\mathsf{in}}, 1, x^{\text{lat}}).$$

and outputting $\mathsf{out}_{id} = (P \cdot Z) \mod 2$.

Note that the joint probability function $\Phi(x)$ is of the form $\Phi(x) = \frac{1}{Z} \prod_{B \in \mathcal{B}'} \psi'_B(x_B)$.

Proof. By the definition of Φ,

$$P = \sum_{x^{\mathsf{in}}, x^{\text{lat}}} \left(\prod_{B \in \mathcal{B}} \psi_B(x_B) \right) \prod_{j=1}^{d} \mathbf{V}_{S_j^{id_j}}(x_j^{\mathsf{in}})$$

is precisely the probability that $X^{\text{out}} = 1$ when the input variables are independently distributed according to $\mathbf{V}_{S_j^{id_j}}$ and the remaining variables are distributed according to the graphical model.

By the assumed representation of f, for every x^{in} there is exactly one setting of x^{lat} contributing when $x^{\text{out}} = f(x^{\mathsf{in}})$, and no setting contributing otherwise. Hence,

$$P = \sum_{x^{\mathsf{in}}} f(x^{\mathsf{in}}) \prod_{j=1}^{d} \mathbf{V}_{S_j^{id_j}}(x_j^{\mathsf{in}}).$$

Using $\mathbf{V}_{S_j^{id_j}}(x_j^{\mathsf{in}}) = \frac{\mathbf{M}_{S_j^{id_j}}(x_j^{\mathsf{in}})}{\|\mathbf{M}_{S_j^{id_j}}\|_1}$, we obtain

$$P \cdot Z = \sum_{x^{\mathsf{in}}} f(x^{\mathsf{in}}) \prod_{j=1}^{d} \mathbf{M}_{S_j^{id_j}}(x_j^{\mathsf{in}}).$$

The right-hand side is exactly the response of the server corresponding to *id* in the basic Chor et al. protocol. Therefore, $\mathsf{out}_{id} = (P \cdot Z) \bmod 2$, as claimed. □

Theorem 2 implies that every exact inference algorithm yields an implementation of the server's computation. One way to perform inference is to leverage the *primal graph* corresponding to the joint probability function $\Phi(x)$.

Definition 3 (Primal Graph)**.** Suppose a joint probability function $\Phi(x)$ has the factorization

$$\Phi(x) = \frac{1}{Z} \prod_{B \in \mathcal{B}'} \psi'_B(x_B),$$

The *primal graph* of Φ is the undirected graph with one vertex for each variable of x. Two distinct variables x_i and x_j are connected by an edge if and only if there exists some $B \in \mathcal{B}'$ such that $i, j \in B$.

Definition 4 (Tree Decomposition)**.** Let $G = (V, E)$ be an undirected graph. A *tree decomposition* of G is a pair $(T, \{B_t\}_{t \in V(T)})$, where T is a tree and each $B_t \subseteq V$ is called a *bag*, satisfying:

1. For every vertex $v \in V$, the set $\{t \in V(T) : v \in B_t\}$ induces a nonempty connected subtree of T.

2. For every edge $(u, v) \in E$, there exists a node $t \in V(T)$ such that $\{u, v\} \subseteq B_t$.

The *width* of a tree decomposition is

$$\max_{t \in V(T)} |B_t| - 1.$$

The *treewidth* of G is the minimum width over all tree decompositions of G.

The following is a standard consequence of [KF09, Dar11].

Theorem 3. *Assume the primal graph corresponding to Φ has at most s variables $x_1, \ldots, x_s$, each with domain Λ_j, satisfies $|\mathcal{B}'| \leq s$, and has a tree decomposition $(T, \{B_t\}_{t \in V(T)})$. Consider an elimination ordering of width at most w, obtained from the tree decomposition. For each variable x_i, let t_i denote the bag in which x_i is eliminated, and let $A_i = B_{t_i} \cap \{x_i, \ldots, x_s\}$. Then the* variable elimination *algorithm can be used to compute any marginal of Φ with runtime*

$$O\left(\sum_i \prod_{x_j \in A_i} |\Lambda_j|\right).$$

In particular, if every variable has domain size D and the tree decomposition has width w, then $|A_i| \leq w + 1$ for every i, and the total running time is $O(sD^{w+1})$.

We are now ready to state our main corollary:

Corollary 1. *If the primal graph corresponding to Φ has treewidth w, each variable has domain size at most D, and there are at most s variables and s factors, then the server's computation in the PIR protocol can be performed in time $O(sD^{w+1})$.*

Proof. By Theorem 2 the server's task is to compute

$$\Pr[X^{\text{out}} = 1] = \sum_{x^{\mathsf{in}}, x^{\text{lat}}} \Phi(x^{\mathsf{in}}, 1, x^{\text{lat}}).$$

This is precisely the problem of computing a marginal of Φ. By Theorem 3, if the primal graph corresponding to Φ has treewidth w, each variable has domain size at most D, and there are s variables, then variable elimination computes this in time $O(sD^{w+1})$. □

5.2.1 Arithmetic Circuits as Graphical Models

We illustrate the reduction using an arithmetic formula over $\mathbb{Z}_q$, where $q = N^{1/d}$.

The input variables correspond to the circuit inputs, the latent variables correspond to wire values, and the output variable corresponds to the output bit. Thus, $x = x^{\mathsf{in}}||x^{\mathsf{out}}||x^{\mathsf{lat}}$ and we denote the output wire by x^{lat}_{out} (i.e. the index of the output wire is *out*).

Each gate gives rise to a local factor. For an addition and multiplication gates with input wires j, k and output wire ℓ, define

$$\mathsf{Gate}_+(x, y, z) = \mathbf{1}[z = x + y \bmod q] \qquad \mathsf{Gate}_\times(x, y, z) = \mathbf{1}[z = xy \bmod q].$$

Additive and multiplicative inverses correspond to gates

$$\mathsf{Gate}_-(x, y) = \mathbf{1}[y = -x \bmod q] \qquad \mathsf{Gate}_{(\cdot)^{-1}}(x, y) = \begin{cases} \mathbf{1}[y = x^{-1} \bmod q] & \text{if } x \neq 0. \\ \mathbf{1}[y = 0 \bmod q] & \text{if } x = 0. \end{cases}$$

Equality constraints are represented by $\mathsf{Eq}(x, y) = \mathbf{1}[x = y]$.

The output constraint checks that the t-th bit of the output wire is equal to the final output of the circuit. $\mathsf{Out}(x, y) = \mathbf{1}[x[t] = y]$.

Let $\mathcal{G}_+$, $\mathcal{G}_\times$, $\mathcal{G}_-$, $\mathcal{G}_{(\cdot)^{-1}}$, consist of the set of tuples (j, k, ℓ) or (j, k) corresponding to the indices of the input and output wires of addition, multiplication, negation, and inverse gates. The resulting factorization of the joint probability function is

$$\Phi(x) = \prod_{(j,k,\ell)\in\mathcal{G}_+} \mathsf{Gate}_+(x^{\mathsf{lat}}_j, x^{\mathsf{lat}}_k, x^{\mathsf{lat}}_\ell) \prod_{(j,k,\ell)\in\mathcal{G}_\times} \mathsf{Gate}_\times(x^{\mathsf{lat}}_j, x^{\mathsf{lat}}_k, x^{\mathsf{lat}}_\ell) \prod_{(j,k)\in\mathcal{G}_-} \mathsf{Gate}_-(x^{\mathsf{lat}}_j, x^{\mathsf{lat}}_k)$$
$$\prod_{(j,k)\in\mathcal{G}_{(\cdot)^{-1}}} \mathsf{Gate}_{(\cdot)^{-1}}(x^{\mathsf{lat}}_j, x^{\mathsf{lat}}_k) \prod_{j=1}^{d} \mathsf{Eq}(x^{\mathsf{in}}_j, x^{\mathrm{w}}_j)\ \mathsf{Out}(x^{\mathrm{out}}, x^{\mathsf{lat}}_{out}) \prod_{j=1}^{d} \mathbf{V}_{S^{id_j}_j}(x^{\mathsf{in}}_j).$$

We can now apply Corollary 1 to conclude that if the primal graph corresponding to Φ has treewidth w, then the server's computation in the PIR protocol can be performed in time $O(sN^{(w+1)/d})$, where s is the size of the circuit. A speedup is obtained when $sN^{(w+1)/d} \in o(N)$.

5.3 Functions of Low Communication Complexity

As another application of our probabilistic reformulation, we consider Boolean functions with low communication complexity.

Theorem 4. *Let $N^{1/d} = 2^\kappa$ and so we can view $x_i \in [N^{1/d}]$ as a κ-length bit-string. Suppose that the Boolean function $f : \{0,1\}^{\kappa\cdot d} \to \{0,1\}$ admits a deterministic d-party communication protocol of cost ρ. Then the server's computation in the PIR protocol can be performed in time $O(2^\rho N^{1/d})$.*

Proof. Suppose that the Boolean function $f : \{0,1\}^{\kappa\cdot d} \to \{0,1\}$ admits a deterministic d-party communication protocol of cost ρ. Equivalently, the domain of f can be partitioned into at most 2^ρ monochromatic rectangles.

Let $\mathcal{R} = \{R_1, \ldots, R_t\}, t \leq 2^\rho$, denote the monochromatic rectangle partition induced by the communication protocol. Each rectangle has the form $R_i = A^1_i \times \cdots \times A^d_i$, where $A^k_i \subseteq [N^{1/d}]$. Moreover, every rectangle is monochromatic, so there exists $c_i \in \{0,1\}$ such that $f(x) = c_i$ for every $x \in R_i$.

For every rectangle R_i with $c_i = 1$, server *id* computes

$$n^k_i = |A^k_i \cap S^{id_k}_k|, \qquad k \in [d]_+.$$

Note that representing the sets A_i^k and $S_k^{id_k}$ by dimension $N^{1/d}$-indicator vectors, the above quantity corresponds to a dot product of the two binary vectors, which can be computed in time $N^{1/d}$.

The quantity $\prod_{k=1}^{d} n_i^k$ is exactly the number of points of R_i contained in $S_1^{id_1} \times \cdots \times S_d^{id_d}$. The server accumulates $T = \sum_{i:c_i=1} \prod_{k=1}^{d} n_i^k$, and outputs $\mathsf{out}_{id} = T \bmod 2$. □

Theorem 4 yields an asymptotic speedup over explicit evaluation of f whenever $2^\rho \in o(N^{1-1/d})$.

5.3.1 Communication Complexity as a Treewidth-One Graphical Model

The construction above can be viewed as a special case of the graphical model framework.

Recall that $N^{1/d} = 2^\kappa$ and we consider $f : \{0,1\}^{\kappa \cdot d} \to \{0,1\}$ that has deterministic d-party communication complexity ρ. As before, there exists a partition of $\{0,1\}^{\kappa \cdot d}$ into at most 2^ρ monochromatic rectangles $R_1, \ldots, R_t, t \leq 2^\rho$, where each rectangle has the form $R_i = A_i^1 \times \cdots \times A_i^d$, and f is constant on every rectangle.

Introduce a latent variable $X^{\mathsf{lat}} \in [t]$, whose value, x^{lat}, specifies the rectangle containing the current input. The graphical model thus contains the variables $X = X^{\mathsf{in}} || X^{\mathsf{out}} || X^{\mathsf{lat}}$, where $X^{\mathsf{in}} = X_1^{\mathsf{in}}, \ldots, X_d^{\mathsf{in}}$ are the input variables and X^{out} is the output bit.

For every input variable X_j^{in}, define the factor $\psi_j(X^{\mathsf{lat}}, X_j^{\mathsf{in}}) = \mathbf{1}[X_j^{\mathsf{in}} \in A^j_{X^{\mathsf{lat}}}]$. Similarly, define $\psi_{\text{out}}(X^{\mathsf{lat}}, X^{\mathsf{out}}) = \mathbf{1}[X^{\mathsf{out}} = c_{X^{\mathsf{lat}}}]$, where $c_{X^{\mathsf{lat}}} \in \{0,1\}$ denotes the monochromatic value of rectangle X^{lat}. The resulting factorization of the unnormalized graphical model is:

$$g(X^{\mathsf{in}} || X^{\mathsf{out}} || X^{\mathsf{lat}}) = \psi_{\text{out}}(X^{\mathsf{lat}}, X^{\mathsf{out}}) \prod_{j=1}^{d} \psi_j(X^{\mathsf{lat}}, X_j^{\mathsf{in}}).$$

Recall that in the PIR setting, we additionally introduce the unary indicator factors $\mathbf{V}_{S_j^{id_j}}(X_j^{\mathsf{in}})$, which restrict each input variable to the subset chosen by the client. We obtain the following factorization of the joint probability function:

$$\Phi(X^{\mathsf{in}} || X^{\mathsf{out}} || X^{\mathsf{lat}}) = \psi_{\text{out}}(X^{\mathsf{lat}}, X^{\mathsf{out}}) \prod_{j=1}^{d} \psi_j(X^{\mathsf{lat}}, X_j^{\mathsf{in}}) \mathbf{V}_{S_j^{id_j}}(X_j^{\mathsf{in}}).$$

Observe that every factor involves either a single variable or involves the latent variable X^{lat} and a single additional variable X_j^{in}. Consequently, the primal graph is a star centered at X^{lat}, and is a tree with leaves corresponding to the variables $X^{\mathsf{in}} || X^{\mathsf{out}}$ and root corresponding to X^{lat}. Using variable elimination from the leaves to the root we therefore obtain that the server's computation:

$$\begin{aligned}
\sum_{x^{\mathsf{in}}, x^{\mathsf{lat}}} \Phi(x^{\mathsf{in}} || 1 || x^{\mathsf{lat}}) &= \sum_{x^{\mathsf{in}}, x^{\mathsf{lat}}} \psi_{\text{out}}(x^{\mathsf{lat}}, 1) \prod_{j=1}^{d} \psi_j(x^{\mathsf{lat}}, x_j^{\mathsf{in}}) \mathbf{V}_{S_j^{id_j}}(x_j^{\mathsf{in}}) \\
&= \sum_{x^{\mathsf{lat}}} \psi_{\text{out}}(x^{\mathsf{lat}}, 1) \left(\sum_{x^{\mathsf{in}}} \prod_{j=1}^{d} \psi_j(x^{\mathsf{lat}}, x_j^{\mathsf{in}}) \mathbf{V}_{S_j^{id_j}}(x_j^{\mathsf{in}}) \right) \\
&= \sum_{x^{\mathsf{lat}}} \psi_{\text{out}}(x^{\mathsf{lat}}, 1) \prod_{j=1}^{d} \left(\sum_{x_j^{\mathsf{in}}} \psi_j(x^{\mathsf{lat}}, x_j^{\mathsf{in}}) \mathbf{V}_{S_j^{id_j}}(x_j^{\mathsf{in}}) \right).
\end{aligned}$$

can be performed in time $O(N^{1/d} 2^\rho)$.

Note that since the inner summation is simply $\frac{|A^j_{x^{\mathsf{lat}}} \cap S_j^{id_j}|}{|S_j^{id_j}|}$, this expression becomes

$$\frac{1}{Z}\sum_{x^{\mathsf{lat}}:c_{x^{\mathsf{lat}}}=1}\prod_{j=1}^{d}|A^{j}_{x^{\mathsf{lat}}}\cap S_j^{id_j}|,$$

where $Z=\prod_{j=1}^{d}|S_j^{\mathrm{id}_j}|$. Multiplying by Z, therefore, yields exactly the result of the previous section. Thus, the communication-complexity algorithm is precisely variable elimination on a treewidth-one graphical model whose latent variable encodes the monochromatic rectangle containing the input.

5.4 Sparse Fourier Representation

We next consider functions that admit sparse Fourier representations. In this case we consider a $-1/1$ function f derived from the Boolean function f_{bool} determining the database by defining $f(x):=2f_{\mathrm{bool}}(x)-1$. The key observation is that since the $-1/1$ output of f can be expressed as a summation of a small number of Fourier characters over the reals, then the server can compute the number of 1-outputs for each character separately and then combine the results.

Theorem 5. *Let $N^{1/d}=2^{\kappa}$ and so we can view $x_i\in[N^{1/d}]$ as a κ-length bit-string. Suppose that the function $f:[N^{1/d}]^d\to\{-1,1\}$ admits an s-sparse Fourier representation. Then the server's computation in the PIR protocol can be performed in time $O(sN^{1/d})$.*

Proof. Let $f(x)=\sum_{\chi}\widehat{f}(\chi)\chi(x)$ be the Fourier expansion of f, where each character $\chi:[N^{1/d}]^d\to\{-1,1\}$ is a parity function. For a fixed character χ, define

$$T_\chi=\left|\left\{x\in S_1^{id_1}\times\cdots\times S_d^{id_d}:\chi(x)=1\right\}\right|.$$

Since a parity function has a d-bit, d-party communication protocol, the low-communication-complexity algorithm of the previous section computes T_χ efficiently.

Let $Z=\prod_{j=1}^{d}|S_j^{id_j}|$ denote the size of the Cartesian product. Instead of counting only the inputs where $\chi(x)=1$, it is convenient to consider the signed sum

$$T_\chi^{\mathrm{diff}}=\sum_x\chi(x)=2T_\chi-Z.$$

By linearity of the Fourier representation,

$$T^{\mathrm{diff}}=\sum_x f(x)=\sum_\chi\widehat{f}(\chi)T_\chi^{\mathrm{diff}}.$$

Therefore, the number of inputs in the Cartesian product on which f evaluates to 1 is $T=\frac{T^{\mathrm{diff}}+Z}{2}$. The server outputs $\mathsf{out}_{id}=T \bmod 2$.

□

Theorem 5 yields an asymptotic speedup over explicit evaluation of f whenever $s=|\{\chi:\widehat{f}(\chi)\neq 0\}|\in o(N^{(d-1)/d})$.

6 The Ambainis Protocol as Probabilistic Inference

The Ambainis protocol recursively invokes a two-server PIR protocol. Theorem 6 shows that these recursive queries continue to induce *product distributions* that can be computed efficiently from the distributions induced by the prior recursive queries. Consequently, all of the inference algorithms developed in the previous sections apply without modification. We begin by re-stating the Ambainis protocol in Section 6.1. We then prove our theorem in Section 6.2.

6.1 The Recursive Construction of Ambainis [Amb97]

We describe the construction of a $\tilde{d}$-server PIR protocol from a $\tilde{d}-1$-server PIR protocol and a 2-server PIR protocol. Let $d' \leq 2\tilde{d}-1$

- The client has an input
$$(x_1^*, \ldots, x_{d'}^*) \in [N^{1/d}]^{d'}.$$
The client first runs the query generation algorithm of the 2-server PIR protocol on the desired index. This produces two queries $q^{(0)}, q^{(1)}$.
Equivalently, the query defines subsets $S_1^0, \ldots, S_{d'}^0 \subseteq [N^{1/d}]$ chosen uniformly at random, together with subsets $S_i^1 = S_i^0 \oplus x_i^*$ for every $i \in [d']_+$.
The first query is sent to the first server while the second query is sent to the remaining $\tilde{d}-1$ servers.

- The first server responds to the query as follows: For every $id \in \{0,1\}^{d'}$ with Hamming weight at most 1, let $S_{id,0}$ denote the set of indices i such that $id = 1$. The first server computes and sends to the client the values:
$$\begin{aligned}\mathsf{out}_{id}[[\tilde{x}_i]_{i\in S_{id,0}}] &= \sum_{\substack{x_i \in S_i^0 \oplus \tilde{x}_i \\ i \in S_{id,0}}} \sum_{\substack{x_i \in S_i^0 \\ i \notin S_{id,0}}} f(x_1, \ldots, x_d) \\ &= \sum_x f(x) \prod_{i \notin S_{id,0}} \mathbf{M}_{S_i^0}[x_i] \cdot \prod_{i \in S_{id,0}} \mathbf{M}_{S_i^0 \oplus \tilde{x}_i}[x_i],\end{aligned}$$
where for a set S, $\mathbf{M}_S$ is the indicator vector for that set.
We call these outputs the *output of virtual server id for index* $[\tilde{x}_i]_{i\in S_{id,0}}$ and denote them by $\mathsf{out}_{id,q^0}([\tilde{x}_i]_{i\in S_{id,0}})$.

- Upon receiving the second query, each of the latter $\tilde{d}-1$ servers constructs an identical collection of auxiliary databases; the subsequent recursive PIR protocol is then executed over these databases.
In particular, for every $id \in \{0,1\}^{d'}$ with Hamming weight at least 2, the servers construct a database whose entries are indexed by tuples
$$(\tilde{x}_i)_{i\in S_{id,1}} \in [N^{1/d}]^{|S_{id,1}|},$$
where $S_{id,1}$ is the set of indices i at which $id_i = 0$.
The entry corresponding to $(\tilde{x}_i)_{i\in S_{id,1}}$ is defined as follows:
$$\begin{aligned}\mathsf{out}_{id}[[\tilde{x}_i]_{i\in S_{id,1}}] &= \sum_{\substack{x_i \in S_i^1 \oplus \tilde{x}_i \\ i \in S_{id,1}}} \sum_{\substack{x_i \in S_i^1 \\ i \notin S_{id,1}}} f(x_1, \ldots, x_d) \\ &= \sum_x f(x) \prod_{i \notin S_{id,1}} \mathbf{M}_{S_i^1}[x_i] \cdot \prod_{i \in S_{id,1}} \mathbf{M}_{S_i^1 \oplus \tilde{x}_i}[x_i],\end{aligned}$$
where for a set S, $\mathbf{M}_S$ is the indicator vector for that set.
We call these outputs the *output of virtual server id for index* $[\tilde{x}_i]_{i\in S_{id,1}}$ and denote them by $\mathsf{out}_{id,q^1}([\tilde{x}_i]_{i\in S_{id,1}})$.
Note that virtual server id's in the next level of the recursion will be bit strings of length $d'' = |S_{id,1}|$, where $d'' \leq d' - 2 \leq 2(\tilde{d}-1) - 1$.

- The client and the latter $\tilde{d}-1$ servers now execute the $(\tilde{d}-1)$-server PIR protocol to retrieve the indices required from the out_{id} databases. Specifically, for each $id = id_1, \ldots, id_{d'} \in \{0,1\}^{d'}$ of Hamming weight at least 2, let $S_{id,1}$ denote the set of indices i for which $id = 0$. The client requests index $[x_i^*]_{i \in S_{id,1}}$ from database out_{id}.
- From the response of the first server, the client can compute $\mathsf{out}_{id} = \mathsf{out}_{id}[[x_i^*]_{i \in S_{id,0}}]$, for all id with Hamming weight at most 1.
- The client receives the answers from the $(\tilde{d}-1)$-server PIR execution and computes $\mathsf{out}_{id} = \mathsf{out}_{id}[[x_i^*]_{i \in S_{id,1}}]$ for all id with Hamming weight at least 2.
- Finally, the client outputs

$$\mathsf{out} = \bigoplus_{id \in \{0,1\}^d} \mathsf{out}_{id}.$$

6.2 Probabilistic Inference Perspective for Ambainis's Protocol

Theorem 6. *Fix a recursion depth $\ell \geq 1$, and let $\mathbf{q}^{(1)} = (q_1^{(1)}, \ldots, q_{\ell-1}^{(1)})$ denote the sequence of second queries from the preceding recursive levels. For $b \in \{0,1\}$, let id_ℓ be a virtual server whose Hamming weight is at most 1 when $b = 0$ and at least 2 when $b = 1$. Then the output of virtual server id_ℓ for index $[\tilde{x}_i]_{i \in S_{id_\ell,b}}$ has the form*

$$\mathsf{out}_{id_\ell, \mathbf{q}^{(1)}, q_\ell^{(b)}}\left([\tilde{x}_i]_{i \in S_{id_\ell,b}}\right) = \left(\sum_{x \in [N^{1/d}]^d} f(x) \prod_{i \notin S_{id_\ell,b}} \mathbf{M}^i_{\mathbf{q}^{(1)}, q_\ell^{(b)}}[x_i] \prod_{i \in S_{id_\ell,b}} \mathbf{M}^i_{\mathbf{q}^{(1)}, q_\ell^{(b)}}(\tilde{x}_i)[x_i] \right) \bmod 2.$$

The vectors

$$\mathbf{M}^i_{\mathbf{q}^{(1)}, q_\ell^{(b)}}, \qquad \mathbf{M}^i_{\mathbf{q}^{(1)}, q_\ell^{(b)}}(\tilde{x}_i)$$

are 0/1 vectors. Furthermore, for every $i \in S_{id_\ell,b}$, every pair $\tilde{x}_i, \tilde{x}_i' \in [N^{1/d}]$, and every $x_i \notin \{\tilde{x}_i, \tilde{x}_i'\}$,

$$\mathbf{M}^i_{\mathbf{q}^{(1)}, q_\ell^{(b)}}(\tilde{x}_i)[x_i] = \mathbf{M}^i_{\mathbf{q}^{(1)}, q_\ell^{(b)}}(\tilde{x}_i')[x_i].$$

The latter property is needed since it allows (implicit) construction of $[\mathbf{M}^i_{\mathbf{q}^{(1)}, q_\ell^{(b)}}(\tilde{x}_i)]_{\tilde{x}_i \in [N^{1/d}]}$ in time $O(N^{1/d})$ from the vectors associated with the previous recursive query.

Corollary 2. *Under the conditions of Theorem 6, if any of the indicator vectors $\mathbf{M}^i_{\mathbf{q}^{(1)}, q_\ell^{(b)}}$, $\mathbf{M}^i_{\mathbf{q}^{(1)}, q_\ell^{(b)}}(\tilde{x}_i)$, for $i \in S_{id_\ell,b}$ are identically zero, the corresponding virtual-server output on input $[\tilde{x}_i]_{i \in S_{id_\ell,b}}$ is zero. Otherwise, normalize the vectors as follows. Define*

$$\mathbf{V}^i_{\mathbf{q}^{(1)}, q_\ell^{(b)}} = \frac{\mathbf{M}^i_{\mathbf{q}^{(1)}, q_\ell^{(b)}}}{\|\mathbf{M}^i_{\mathbf{q}^{(1)}, q_\ell^{(b)}}\|_1}, i \notin S_{id_\ell,b}, \quad \text{and} \quad \mathbf{V}^i_{\mathbf{q}^{(1)}, q_\ell^{(b)}}(\tilde{x}_i) = \frac{\mathbf{M}^i_{\mathbf{q}^{(1)}, q_\ell^{(b)}}(\tilde{x}_i)}{\|\mathbf{M}^i_{\mathbf{q}^{(1)}, q_\ell^{(b)}}(\tilde{x}_i)\|_1}, i \in S_{id_\ell,b}.$$

Then the output of each virtual server is obtained by evaluating the corresponding marginal probability under a product distribution

$$\Pr_{\substack{X_i \sim \mathbf{V}^i_{\mathbf{q}^{(1)}, q_\ell^{(b)}}, i \notin S_{id_\ell,b} \\ X_i \sim \mathbf{V}^i_{\mathbf{q}^{(1)}, q_\ell^{(b)}}(\tilde{x}_i), i \in S_{id_\ell,b}}} [f(X) = 1] = \sum_{x \in [N^{1/d}]^d} f(x) \prod_{i \notin S_{id_\ell,b}} \mathbf{V}^i_{\mathbf{q}^{(1)}, q_\ell^{(b)}}[x_i] \prod_{i \in S_{id_\ell,b}} \mathbf{V}^i_{\mathbf{q}^{(1)}, q_\ell^{(b)}}(\tilde{x}_i)[x_i],$$

followed by multiplication by the normalization factor

$$Z = \prod_{i \notin S_{id_\ell,b}} \|\mathbf{M}^i_{\mathbf{q}^{(1)}, q_\ell^{(b)}}\|_1 \prod_{i \in S_{id_\ell,b}} \|\mathbf{M}^i_{\mathbf{q}^{(1)}, q_\ell^{(b)}}(\tilde{x}_i)\|_1$$

and taking modulo 2.

Corollary 2 immediately implies that all of the inference algorithms described in the previous sections continue to apply to the recursive Ambainis protocol.

Proof of Theorem 6. We prove the theorem by induction on the recursion depth.

For recursive depth ℓ, let $id_\ell \in \{0,1\}^{d'}$ for some $d' \leq d - 2(\ell-1)$. Messages $\mathsf{out}_{id_\ell,\mathbf{q}^{(1)},q_\ell^0}([\tilde{x}_i]_{i\in S_{id_\ell,0}})$ sent by the ℓ-th server when responding to the ℓ-th recursive query correspond to the outputs of each virtual server id_ℓ with Hamming weight 0 or 1, for index $[\tilde{x}_i]_{S_{id_\ell,0}}$.

Values $\mathsf{out}_{id_\ell,\mathbf{q}^{(1)},q_\ell^{(1)}}([\tilde{x}_i]_{i\in S_{id_\ell,1}})$ computed by servers $\ell+1$ through $\tilde{d}$ as entries of Table id_ℓ during the ℓ-th recursive query correspond to the outputs of each virtual server id_ℓ with Hamming weight 2 or more, for index $[\tilde{x}_i]_{S_{id_\ell},1}$.

We first consider the base case $\ell = 1$. By the definition of the basic Chor et al protocol, for $b \in \{0,1\}$:

$$\mathsf{out}_{id_1,q_1^{(b)}}([\tilde{x}_i]_{i\in S_{id_1,b}}) = \sum_{x\in[N^{1/d}]^d} f(x) \prod_{i\notin S_{id_1,b}} \mathbf{M}_{S_1^b}[x_i] \prod_{i\in S_{id_1,b}} \mathbf{M}_{S_1^b\oplus\tilde{x}_i}[x_i].$$

Note that for every pair $(\tilde{x}_i,\tilde{x}_i')$ and every $x_i \notin \{\tilde{x}_i,\tilde{x}_i'\}$, if $x_i \in S_1^b$ then $\mathbf{M}_{S_1^b\oplus\tilde{x}_i}[x_i] = 1 = \mathbf{M}_{S_1^b\oplus\tilde{x}_i'}[x_i]$ and if $x_i \notin S_1^b$ then $\mathbf{M}_{S_1^b\oplus\tilde{x}_i}[x_i] = 0 = \mathbf{M}_{S_1^b\oplus\tilde{x}_i'}[x_i]$. Thus, the base case is immediate by setting $\mathbf{M}^i_{q_1^b} := \mathbf{M}_{S_1^b}$ for $i \notin S_{id_1,b}$ and $\mathbf{M}^i_{q_1^b}(\tilde{x}_i) := \mathbf{M}_{S_1^b\oplus\tilde{x}_i}$ for $i \in S_{id_1,b}$.

We now prove the inductive step. Specifically, assuming that the theorem holds at recursion depth ℓ, we show that after applying the recursive query at level $\ell+1$, the resulting virtual-server output again has the required form, and that the newly defined vectors satisfy the same consistency property. Details follow.

Assume the theorem holds after ℓ recursive levels. Thus, for each id_ℓ

$$\mathsf{out}_{id_\ell,\mathbf{q}^{(1)},q_\ell^{(b)}}([\tilde{x}_i]_{i\in S_{id_\ell,b}}) = \left(\sum_{x\in[N^{1/d}]^d} f(x) \prod_{i\notin S_{id_\ell,b}} \mathbf{M}^i_{\mathbf{q}^{(1)},q_\ell^{(b)}}[x_i] \prod_{i\in S_{id_\ell,b}} \mathbf{M}^i_{\mathbf{q}^{(1)},q_\ell^{(b)}}(\tilde{x}_i)[x_i]\right) \bmod 2,$$

where $b = 0$ for id_ℓ of Hamming weight at most 1 and $b = 1$ for id_ℓ of Hamming weight at least 2.

Now consider a recursive query $(q_{\ell+1}^{(0)}, q_{\ell+1}^{(1)})$, to database id_ℓ where for $b \in \{0,1\}$,

$$q_{\ell+1}^{(b)} = (id_\ell, [S^b_{\ell+1,i}]_{i\in S_{id_\ell,1}}),$$

The output of virtual server $id_{\ell+1}$ for index $[\hat{x}_i]_{i\in S_{id_{\ell+1},b}}$ corresponds to the following, where $b \in \{0,1\}$ is set appropriately depending on the Hamming weight of $id_{\ell+1}$.

$$\begin{aligned}
&\mathsf{out}_{id_{\ell+1},\mathbf{q}^1,q_{\ell+1}^{(b)}}([\hat{x}_i]_{i\in S_{id_{\ell+1},b}}) \\
&= \sum_{\substack{\tilde{x}_i \\ i\in S_{id_\ell,1}}} \left(\sum_{x\in[N^{1/d}]^d} f(x) \prod_{i\notin S_{id_\ell,1}} \mathbf{M}^i_{\mathbf{q}^{(1)}}[x_i] \prod_{i\in S_{id_\ell,1}} \mathbf{M}^i_{\mathbf{q}^{(1)}}(\tilde{x}_i)[x_i]\right) \\
&\qquad\qquad \cdot \prod_{i\in S_{id_\ell,1}\setminus S_{id_{\ell+1},b}} \mathbf{M}_{S^b_{\ell+1,i}}[\tilde{x}_i] \prod_{i\in S_{id_{\ell+1},b}} \mathbf{M}_{S^b_{\ell+1,i}\oplus\hat{x}_i}[\tilde{x}_i] \bmod 2 \\
&= \sum_{\substack{\tilde{x}_i \\ i\in S_{id_\ell,1}}} \sum_{x\in[N^{1/d}]^d} f(x) \prod_{i\notin S_{id_\ell,1}} \mathbf{M}^i_{\mathbf{q}^{(1)}}[x_i] \prod_{i\in S_{id_\ell,1}} \mathbf{M}^i_{\mathbf{q}^{(1)}}(\tilde{x}_i)[x_i] \\
&\qquad\qquad \cdot \prod_{i\in S_{id_\ell,1}\setminus S_{id_{\ell+1},b}} \mathbf{M}_{S^b_{\ell+1,i}}[\tilde{x}_i] \prod_{i\in S_{id_{\ell+1},b}} \mathbf{M}_{S^b_{\ell+1,i}\oplus\hat{x}_i}[\tilde{x}_i] \bmod 2
\end{aligned}$$

$$= \sum_{\substack{\tilde{x}_i \\ i \in S_{id_\ell,1}}} \sum_{x \in [N^{1/d}]^d} f(x) \prod_{i \notin S_{id_\ell,1}} \mathbf{M}^i_{\mathbf{q}^{(1)}}[x_i] \prod_{i \in S_{id_\ell,1} \setminus S_{id_{\ell+1},b}} \mathbf{M}^i_{\mathbf{q}^{(1)}}(\tilde{x}_i)[x_i] \mathbf{M}_{S^b_{\ell+1,i}}[\tilde{x}_i]$$

$$\cdot \prod_{i \in S_{id_{\ell+1},b}} \mathbf{M}^i_{\mathbf{q}^{(1)}_\ell}(\tilde{x}_i)[x_i]\, \mathbf{M}_{S^b_{\ell+1,i} \oplus \hat{x}_i}[\tilde{x}_i] \bmod 2$$

$$= \sum_{x \in [N^{1/d}]^d} f(x) \prod_{i \notin S_{id_\ell,1}} \mathbf{M}^i_{\mathbf{q}^{(1)}}[x_i]$$

$$\cdot \sum_{\substack{\tilde{x}_i \\ i \in S_{id_\ell,1}}} \prod_{i \in S_{id_\ell,1} \setminus S_{id_{\ell+1},b}} \mathbf{M}^i_{\mathbf{q}^{(1)}}(\tilde{x}_i)[x_i] \mathbf{M}_{S^b_{\ell+1,i}}[\tilde{x}_i] \prod_{i \in S_{id_{\ell+1},b}} \mathbf{M}^i_{\mathbf{q}^{(1)}}(\tilde{x}_i)[x_i]\, \mathbf{M}_{S^b_{\ell+1,i} \oplus \hat{x}_i}[\tilde{x}_i] \bmod 2$$

$$= \sum_{x \in [N^{1/d}]^d} f(x) \prod_{i \notin S_{id_\ell,1}} \mathbf{M}^i_{\mathbf{q}^{(1)}}[x_i]$$

$$\cdot \prod_{i \in S_{id_\ell,1} \setminus S_{id_{\ell+1},b}} \left(\sum_{\tilde{x}_i} \mathbf{M}^i_{\mathbf{q}^{(1)}}(\tilde{x}_i)[x_i] \mathbf{M}_{S^b_{\ell+1,i}}[\tilde{x}_i] \right) \prod_{i \in S_{id_{\ell+1},b}} \left(\sum_{\tilde{x}_i} \mathbf{M}^i_{\mathbf{q}^{(1)}}(\tilde{x}_i)[x_i]\, \mathbf{M}_{S^b_{\ell+1,i} \oplus \hat{x}_i}[\tilde{x}_i] \right) \bmod 2,$$

where the last equality follows since the summation over $[\tilde{x}_i]_{i \in S_{id_\ell,1}}$ factors across the coordinates. For $b \in \{0,1\}$, we next define the vectors $[\mathbf{M}^i_{\mathbf{q}^{(1)},q^{(b)}_{\ell+1}}]_{i \notin S_{id_\ell,1}}$, $[\mathbf{M}^i_{\mathbf{q}^{(1)},q^{(b)}_{\ell+1}}]_{i \in S_{id_\ell,1} \setminus S_{id_{\ell+1}},b}$ and $[\mathbf{M}^i_{\mathbf{q}^{(1)},q^{(b)}_{\ell+1}}(\hat{x}_i)]_{i \in S_{id_{\ell+1},b}}$:

For $i \notin S_{id_\ell,1}$: Let

$$\mathbf{M}^i_{\mathbf{q}^{(1)},q^{(b)}_{\ell+1}} := \mathbf{M}^i_{\mathbf{q}^{(1)}}.$$

For $i \in S_{id_\ell,1} \setminus S_{id_{\ell+1},b}$: Let

$$\mathbf{M}^i_{\mathbf{q}^{(1)},q^{(b)}_{\ell+1}}[x_i] := \sum_{\tilde{x}_i} \mathbf{M}^i_{\mathbf{q}^{(1)}}(\tilde{x}_i)[x_i] \mathbf{M}_{S^b_{\ell+1,i}}[\tilde{x}_i] \bmod 2.$$

Using the inductive hypothesis, we have that for all pairs $(\tilde{x}_i, \tilde{x}'_i)$, and $x_i \notin \{\tilde{x}_i, \tilde{x}'_i\}$ $\mathbf{M}^i_{\mathbf{q}^{(1)}}(\tilde{x}_i)[x_i] = \mathbf{M}^i_{\mathbf{q}^{(1)}}(\tilde{x}'_i)[x_i]$. Let $\mathbf{M}^i_{\mathbf{q}^{(1)}}(*)[x_i]$ denote this value.

So for all $x_i \in [N^{1/d}]$,

$$\mathbf{M}^i_{\mathbf{q}^{(1)},q^{(b)}_{\ell+1}}[x_i] = \sum_{\tilde{x}_i} \mathbf{M}^i_{\mathbf{q}^{(1)}}(\tilde{x}_i)[x_i] \mathbf{M}_{S^b_{\ell+1,i}}[\tilde{x}_i] \bmod 2$$

$$= \begin{cases} |S^b_{\ell+1,i}| \cdot \mathbf{M}^i_{\mathbf{q}^{(1)}}(*)[x_i] \bmod 2 & \text{if } x_i \notin S^b_{\ell+1,i} \\ (|S^b_{\ell+1,i}| - 1) \cdot \mathbf{M}^i_{\mathbf{q}^{(1)}}(*)[x_i] + \mathbf{M}^i_{\mathbf{q}^{(1)}}(x_i)[x_i] \bmod 2, & \text{if } x_i \in S^b_{\ell+1,i}. \end{cases}$$

The above implies that $\mathbf{M}^i_{\mathbf{q}^{(1)},q^{(b)}_{\ell+1}}$ can be computed in time $O(N^{1/d})$ given the implicit representation of $[\mathbf{M}^i_{\mathbf{q}^{(1)}}(\tilde{x}_i)]_{\tilde{x}_i}$.

For $i \in S_{id_{\ell+1},b}$: For all $x_i \in [N^1/d]$, let

$$\mathbf{M}^i_{\mathbf{q}^{(1)},q^{(b)}_{\ell+1}}(\hat{x}_i)[x_i] := \sum_{\tilde{x}_i} \mathbf{M}^i_{\mathbf{q}^{(1)}}(\tilde{x}_i)[x_i] \mathbf{M}_{S^b_{\ell+1,i} \oplus \hat{x}_i}[\tilde{x}_i] \bmod 2$$

Invoking the inductive hypothesis and the definition of $\mathbf{M}_{S^b_{\ell+1,i} \oplus \hat{x}_i}$, we have that

$$\mathbf{M}^i_{\mathbf{q}^{(1)},q^{(b)}_{\ell+1}}(\hat{x}_i)[x_i] = \sum_{\tilde{x}_i} \mathbf{M}^i_{\mathbf{q}^{(1)}}(\tilde{x}_i)[x_i] \mathbf{M}_{S^b_{\ell+1,i} \oplus \hat{x}_i}[\tilde{x}_i] \bmod 2$$

$$= \begin{cases} |S^b_{\ell+1,i} \oplus \hat{x}_i| \cdot \mathbf{M}^i_{\mathbf{q}^{(1)}}(*)[x_i] \bmod 2, & \text{if } x_i \notin S^b_{\ell+1,i} \oplus \hat{x}_i \\ (|S^b_{\ell+1,i} \oplus \hat{x}_i| - 1) \cdot \mathbf{M}^i_{\mathbf{q}^{(1)}}(*)[x_i] + \mathbf{M}^i_{\mathbf{q}^{(1)}}(x_i)[x_i] \bmod 2, & \text{if } x_i \in S^b_{\ell+1,i} \oplus \hat{x}_i \end{cases}$$

The above implies that all of $[\mathbf{M}^i_{\mathbf{q}^{(1)},q^{(b)}_{\ell+1}}(\hat{x}_i)]_{\hat{x}_i}$ can be implicitly computed in time $O(N^{1/d})$ given the implicit representation of $[\mathbf{M}^i_{\mathbf{q}^{(1)}}(\tilde{x}_i)]_{\tilde{x}_i}$.

We now need to show that the $\mathbf{M}^i_{\mathbf{q}^{(1)},q^{(b)}_{\ell+1}}(\hat{x}_i)$ vectors satisfy the property that for any pair $(\hat{x}_i, \hat{x}'_i)$, if $x_i \notin \{\hat{x}_i, \hat{x}'_i\}$ then $\mathbf{M}^i_{\mathbf{q}^{(1)},q^{(b)}_{\ell+1}}(\hat{x}_i)[x_i] = \mathbf{M}^i_{\mathbf{q}^{(1)},q^{(b)}_{\ell+1}}(\hat{x}'_i)[x_i]$.

We first observe that for any $\hat{x}_i, \hat{x}'_i$, $|S^b_{\ell+1,i} \oplus \hat{x}_i| = |S^b_{\ell+1,i} \oplus \hat{x}'_i| \bmod 2$. Clearly, if $\hat{x}_i, \hat{x}'_i$ are both either in $S^b_{\ell+1,i}$ or not in $S^b_{\ell+1,i}$ then the claim holds. In the case that one is in the set and one is not, the set sizes differ by exactly 2 and therefore $|S^b_{\ell+1,i} \oplus \hat{x}_i| = |S^b_{\ell+1,i} \oplus \hat{x}'_i| \bmod 2$.

Case 1: $x_i \in S^b_{\ell+1,i}$. In this case, $x_i \in S^b_{\ell+1,i} \oplus \hat{x}_i$ and $x_i \in S^b_{\ell+1,i} \oplus \hat{x}'_i$. Thus $\mathbf{M}^i_{\mathbf{q}^{(1)},q^{(b)}_{\ell+1}}(\hat{x}_i)[x_i] = (|S^b_{\ell+1,i} \oplus \hat{x}_i| - 1) \cdot \mathbf{M}^i_{\mathbf{q}^{(1)}}(*)[x_i] + \mathbf{M}^i_{\mathbf{q}^{(1)}}(x_i)[x_i] \bmod 2 = (|S^b_{\ell+1,i} \oplus \hat{x}'_i| - 1) \cdot \mathbf{M}^i_{\mathbf{q}^{(1)}}(*)[x_i] + \mathbf{M}^i_{\mathbf{q}^{(1)}}(x_i)[x_i] \bmod 2 = \mathbf{M}^i_{\mathbf{q}^{(1)},q^{(b)}_{\ell+1}}(\hat{x}'_i)[x_i]$.

Case 2: $x_i \notin S^b_{\ell+1,i}$. In this case, $x_i \notin S^b_{\ell+1,i} \oplus \hat{x}_i$ and $x_i \notin S^b_{\ell+1,i} \oplus \hat{x}'_i$. Thus $\mathbf{M}^i_{\mathbf{q}^{(1)},q^{(b)}_{\ell+1}}(\hat{x}_i)[x_i] = |S^b_{\ell+1,i} \oplus \hat{x}_i| \cdot \mathbf{M}^i_{\mathbf{q}^{(1)}}(*)[x_i] \bmod 2 = |S^b_{\ell+1,i} \oplus \hat{x}'_i| \cdot \mathbf{M}^i_{\mathbf{q}^{(1)}}(*)[x_i] \bmod 2 = \mathbf{M}^i_{\mathbf{q}^{(1)},q^{(b)}_{\ell+1}}(\hat{x}'_i)[x_i]$.

Thus, the property holds in all cases. This completes the induction. □

Observe that all vectors constructed in the induction are 0/1-valued, since they are defined by sums and products of 0/1 values modulo 2. Therefore, at every recursive level, the resulting representation is a parity sum over a Cartesian product of subsets, which can be computed exactly by evaluating the corresponding product-distribution marginal, multiplying by the normalization factor, and reducing modulo 2.

7 Impossibility Results

In this section we show barriers to reducing the server complexity for all Boolean functions with poly-sized circuits.

Theorem 7. *Let $L \in \mathsf{NP}$, and let $V(x, w)$ be a polynomial-time verifier for L. For every input x, let $\mathbf{V}_x(w) = V(x, w)$ denote the verifier with the instance x hardwired.*

Suppose there exists an implementation of the server algorithm in the Chor et al. PIR protocol that, on every Boolean function represented by $\mathbf{V}_x$, runs in time T.

Then there is a randomized algorithm deciding L in time $O(\mathsf{poly}(|x|) + N^{1/d} + T)$, which always rejects statements not in the language and accepts statements in the language with probability $2/3$.

Proof. Given an instance x, construct the Boolean function $f = \mathbf{V}_x$. The algorithm repeatedly performs the following experiment.

1. Sample the subsets $S^0_1, \ldots, S^0_d$ as in the Chor et al./Ambainis protocol.
2. Send the resulting query to server id **0** in the Chor et al. protocol or the first server in the Ambainis protocol.

3. If server id **0** outputs 1 in the Chor et al. protocol or if the output of virtual server id **0** is 1 in the Ambainis protocol, halt and accept.

After $c2^d \in O(1)$ independent repetitions, reject if no execution returned 1.

If $x \notin L$, then $f \equiv 0$, so the server always returns 0 and the algorithm rejects.

Suppose instead that $x \in L$. Then f is not identically zero. We show next that the server outputs 1 with probability at least 2^{-d}. Let $\mathbf{M}_{S_j}[i]$ denote the indicator variable of the event $i \in S_j$. These variables are mutually independent and uniform random in $\{0,1\}$. The parity computed by the server can be viewed as a function of the variables $[\mathbf{M}_{S_1}[i_1]]_{i_1 \in [N^{1/d}]}, \ldots, [\mathbf{M}_{S_d}[i_d]]_{i_1 \in [N^{1/d}]}$:

$$P([\mathbf{M}_{S_1}[i_1], \ldots, [\mathbf{M}_{S_d}[i_d]]_{i_1,\ldots,i_d \in [N^{1/d}]^d}) = \sum_{(i_1,\ldots,i_d)} f(i_1,\ldots,i_d)\mathbf{M}_{S_1}[i_1]\cdots\mathbf{M}_{S_d}[i_d] \pmod 2.$$

This is a multilinear polynomial over $\mathbb{F}_2$ having $dN^{1/d}$ variables and degree at most d.

Since f is not identically zero, there exists $(i_1,\ldots,i_d)$ such that $f(i_1,\ldots,i_d) = 1$. Setting $\mathbf{M}_{S_j}[i_j] = 1$ for $j \in [d]_+$ and all remaining variables equal to zero makes $P([\mathbf{M}_{S_1}[i_1]]_{i_1 \in [N^{1/d}]}, \ldots, [\mathbf{M}_{S_d}[i_d]]_{i_1 \in [N^{1/d}]}) = 1$. Hence P is not the zero polynomial.

The proof now follows from the following standard fact, which follows from the minimum distance of the Reed-Muller code.

Lemma 2. *Every nonzero multilinear polynomial over $\mathbb{F}_2$ of degree at most d evaluates to 1 on at least a 2^{-d} fraction of its inputs.*

Applying the lemma to P gives $\Pr[P([\mathbf{M}_{S_1}[i_1]]_{i_1 \in [N^{1/d}]}, \ldots, [\mathbf{M}_{S_d}[i_d]]_{i_1 \in [N^{1/d}]}) = 1] \geq 2^{-d}$. Since every execution chooses the setting of the variables $[\mathbf{M}_{S_1}[i_1]]_{i_1 \in [N^{1/d}]}$, $\ldots, [\mathbf{M}_{S_d}[i_d]]_{i_1 \in [N^{1/d}]}$ uniformly at random, every execution accepts with probability at least 2^{-d}. After $c2^d$ independent repetitions, $\Pr[\text{reject}] \leq (1-2^{-d})^{c2^d} \leq e^{-c}$, which is less than $1/3$ for $c = 2$. This completes the proof. □

We next formally define the randomized Circuit Strong Exponential Time Hypothesis (rCSETH), a Circuit-SAT analogue of the Strong Exponential Time Hypothesis (SETH), which is defined with respect to k-SAT. We consider a randomized, polynomial-size-circuit analogue of the Circuit-SAT form of SETH discussed in [Wil24].

Definition 5 (Randomized Circuit Strong Exponential Time Hypothesis (rCSETH))**.** For every constant $\varepsilon > 0$, there exists a constant c such that no uniform randomized algorithm decides satisfiability of all Boolean circuits with n inputs and at most n^c gates in time $O(2^{(1-\varepsilon)n})$ with bounded error.

Setting $N = 2^n$ we obtain the following corollary:

Corollary 3. *Assume rCSETH and let $d > 1$ be a fixed integer. For every $\varepsilon > 0$, there exists a constant c such that no uniform server algorithm for the Chor et al./Ambainis PIR protocols can compute the first (virtual) server's response on every n^c-sized circuit with worst-case running time $O(N^{1-\varepsilon})$, where $N = 2^n$ and n is the number of input variables in the circuit.*

Proof. Suppose, toward a contradiction, that for some $\varepsilon > 0$, and for every c, there exists a uniform server algorithm that computes the first (virtual) server's response on every n^c-sized circuit in time $O(N^{1-\varepsilon})$. By Theorem 7, this yields a randomized algorithm for Circuit-SAT for all circuits of size n^c with running time $O(\mathsf{poly}(|x|)+N^{1/d}+N^{1-\varepsilon})$. Setting $\varepsilon' := \min(\varepsilon, 1-\frac{1}{d})$, we obtain an $O(2^{(1-\varepsilon')n})$-time randomized algorithm, contradicting rCSETH. □

The following corollary is immediate:

Corollary 4. *Assume rCSETH and let $d > 1$ be a fixed integer. For every $\varepsilon > 0$, there exists a constant c such that there is no uniform algorithm that can compute the response of the first (virtual) server in the Chor et al./Ambainis PIR protocols on all Boolean functions $f : \{0,1\}^n \to \{0,1\}$ with circuit size n^c, in time $O(N^{1-\varepsilon})$, where $N = 2^n$ is the size of the database.*

Use of AI Tools

During the preparation of this manuscript, the authors used GPT (OpenAI) as assistive tools for discussion of ideas, preliminary checks, and editing. All mathematical arguments were independently written by the authors, who take full responsibility for the content of the paper.

References

[ACL+21] Sebastian Angel, Hao Chen, Kim Laine, Srinath Setty, et al. Spiral: Fast, high-rate single-server PIR via fhe composition. In *Proceedings of the IEEE Symposium on Security and Privacy (S&P)*. IEEE, 2021.

[ACLS18] Sebastian Angel, Hao Chen, Kim Laine, and Srinath Setty. PIR with compressed queries and amortized query processing. In *Proceedings of the IEEE Symposium on Security and Privacy (S&P)*, pages 962–979. IEEE, 2018.

[Amb97] Andris Ambainis. Upper bound on communication complexity of private information retrieval. In Pierpaolo Degano, Roberto Gorrieri, and Alberto Marchetti-Spaccamela, editors, *ICALP 97*, volume 1256 of *LNCS*, pages 401–407. Springer, Berlin, Heidelberg, July 1997. doi:10.1007/3-540-63165-8_196.

[AMBFK16] Carlos Aguilar-Melchor, Joris Barrier, Laurent Fousse, and Marc-Olivier Killijian. XPIR: Private information retrieval for everyone. In *Network and Distributed System Security Symposium (NDSS)*, 2016.

[Ant26] Anthropic. A $2^{o(n)}$ Time Algorithm for 3-SAT. Hexagon preprint 2610.00577v1, 2026. URL: https://hexagonmath.org/2610.00577v1.

[BCG+23] Elette Boyle, Geoffroy Couteau, Niv Gilboa, Yuval Ishai, et al. SimplePIR: Simple and efficient private information retrieval. In *32nd USENIX Security Symposium (USENIX Security)*. USENIX Association, 2023.

[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 *43rd FOCS*, pages 261–270. IEEE Computer Society Press, November 2002. doi:10.1109/SFCS.2002.1181949.

[CGKS95] Benny Chor, Oded Goldreich, Eyal Kushilevitz, and Madhu Sudan. Private information retrieval. In *36th FOCS*, pages 41–50. IEEE Computer Society Press, October 1995. doi:10.1109/SFCS.1995.492461.

[CGKS98] Benny Chor, Oded Goldreich, Eyal Kushilevitz, and Madhu Sudan. Private information retrieval. *Journal of the ACM*, 45(6):965–982, November 1998. doi:10.1145/293347.293350.

[Dar11] Adnan Darwiche. *Modeling and Reasoning with Bayesian Networks*. Cambridge University Press, 2011.

[DG15] Zeev Dvir and Sivakanth Gopi. 2-server PIR with sub-polynomial communication. In Rocco A. Servedio and Ronitt Rubinfeld, editors, *47th ACM STOC*, pages 577–584. ACM Press, June 2015. doi:10.1145/2746539.2746546.

[Efr09] Klim Efremenko. 3-query locally decodable codes of subexponential length. In Michael Mitzenmacher, editor, *41st ACM STOC*, pages 39–44. ACM Press, May / June 2009. doi:10.1145/1536414.1536422.

[GKS25] Fatemeh Ghasemi, Swastik Kopparty, and Madhu Sudan. Improved PIR schemes using matching vectors and derivatives. In Michal Koucký and Nikhil Bansal, editors, *57th ACM STOC*, pages 1648–1656. ACM Press, June 2025. doi:10.1145/3717823.3718313.

[GR05] Craig Gentry and Zulfikar Ramzan. Single-database private information retrieval with constant communication rate. In Luís Caires, Giuseppe F. Italiano, Luís Monteiro, Catuscia Palamidessi, and Moti Yung, editors, *ICALP 2005*, volume 3580 of *LNCS*, pages 803–815. Springer, Berlin, Heidelberg, July 2005. doi:10.1007/11523468_65.

[KF09] Daphne Koller and Nir Friedman. *Probabilistic Graphical Models: Principles and Techniques*. MIT Press, 2009.

[KO97] Eyal Kushilevitz and Rafail Ostrovsky. Replication is NOT needed: SINGLE database, computationally-private information retrieval. In *38th FOCS*, pages 364–373. IEEE Computer Society Press, October 1997. doi:10.1109/SFCS.1997.646125.

[LS88] Steffen L. Lauritzen and David J. Spiegelhalter. Local computations with probabilities on graphical structures and their application to expert systems. *Journal of the Royal Statistical Society, Series B*, 50(2):157–224, 1988.

[Ope26] OpenAI. The Exponential Time Hypothesis is False. OpenAI Math Release preprint, 2026. URL: https://github.com/openai/math/blob/main/preprints/The-Exponential-Time-Hypothesis-is-False-October-25-2026/paper.pdf.

[PCdG+19] Nathalie Peyrard, M-J Cros, Simon de Givry, Alain Franc, Stephane Robin, Regis Sabbadin, Thomas Schiex, and Matthieu Vignes. Exact or approximate inference in graphical models: why the choice is dictated by the treewidth, and how variable elimination can be exploited. *Australian & New Zealand Journal of Statistics*, 61(2):89–133, 2019.

[Pea88] Judea Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, 1988.

[Wil24] R. Ryan Williams. Self-improvement for circuit-analysis problems. In Bojan Mohar, Igor Shinkar, and Ryan O'Donnell, editors, *56th ACM STOC*, pages 1374–1385. ACM Press, June 2024. doi:10.1145/3618260.3649723.

[Yek07] Sergey Yekhanin. Towards 3-query locally decodable codes of subexponential length. In David S. Johnson and Uriel Feige, editors, *39th ACM STOC*, pages 266–274. ACM Press, June 2007. doi:10.1145/1250790.1250830.