List-Decodable Byzantine Robust PIR: Lower Communication Complexity, Higher Byzantine Tolerance, Smaller List Size

Pengzhen Ke[1], Liang Feng Zhang[1*], Huaxiong Wang[2], and Li-Ping Wang[3]

[1] School of Information Science and Technology, ShanghaiTech University, Shanghai, China
{kepzh,zhanglf}@shanghaitech.edu.cn
[2] School of Physical and Mathematical Sciences, Nanyang Technological University, Singapore
hxwang@ntu.edu.sg
[3] Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China
wangliping@iie.ac.cn

Abstract. Private Information Retrieval (PIR) is a privacy-preserving primitive in cryptography. Significant endeavors have been made to address the variant of PIR concerning the malicious servers. Among those endeavors, list-decodable Byzantine robust PIR schemes may tolerate a majority of malicious responding servers that provide incorrect answers. In this paper, we propose two perfect list-decodable BRPIR schemes. Our schemes are the first ones that can simultaneously handle a majority of malicious responding servers, achieve a communication complexity of $o(n^{1/2})$ for a database of size n, and provide a nontrivial estimation on the list sizes. Compared with the existing solutions, our schemes attain lower communication complexity, higher byzantine tolerance, and smaller list size.

Keywords: Byzantine Robust PIR · List-decoding Algorithms · Malicious Servers · Security.

1 Introduction

Private Information Retrieval (PIR) [10] allows a client to retrieve an element x_i from a database $\mathbf{x} = (x_1, \ldots, x_n)$ without disclosing to the servers which specific element is being accessed. PIR is a fundamental privacy-preserving primitive in cryptography and has widespread applications in systems such as private media browsing [18], metadata-private messaging [3] and location-based services for smartphones [25,39].

The efficiency of a PIR scheme is mainly measured by its *communication complexity*, i.e., the total number of bits that have to be exchanged between the client and all servers in order to retrieve one bit of the database. A *trivial PIR* scheme requires the client to download the entire database from a server. Despite of achieving *perfect* privacy, it incurs a prohibitive communication complexity

that scales linearly in the size n of the database. Chor et al. [10] showed that the trivial PIR scheme is optimal in terms of communication complexity if there is only one server and perfect privacy is required. In order to achieve the nontrivial communication complexity of $o(n)$, one has to either give up the perfect privacy or use multiple servers. One the one hand, by giving up the perfect privacy, single-server PIR schemes [2,30,31,32] with $o(n)$ communication complexity have devised under various computational assumptions. On the other hand, a *multi-server* PIR model [5,13,14,38,40] is necessary if both perfect privacy and $o(n)$ communication complexity are desired. In a k-server PIR scheme, the database $\mathbf{x}$ is replicated among k servers and the client retrieves a database element x_i by querying every server once, such that each individual server learns no information about the retrieval index i. A *t-private* k-server ($t < k$) PIR scheme [38] provides the stronger security guarantee that the retrieval index i is perfectly private for any t colluding servers.

Compared with the single-server PIR model, the multi-server PIR model fundamentally differs across three critical dimensions:

- **Privacy**: Single-server PIR schemes rely on computational hardness assumptions, whereas most of the multi-server PIR achieves *information-theoretic* privacy assuming non-collusion.
- **Computation Complexity**: Multi-server PIR schemes are free of the heavy public-key operations that are required by single-server PIR and thus much faster (possibly by several orders of magnitude).
- **Error Tolerance**: While single-server PIR schemes face complete failure under server compromise, multi-server PIR may still guarantee correct retrieval if some of the servers are controlled maliciously.

These architectural advantages make multi-server PIR an attractive approach, offering stronger security guarantees alongside practical efficiency. However, the involvement of more servers also introduces new challenges. As the number of participating servers increases, so does the risk of receiving incorrect responses—whether due to network failures, outdated data, or deliberate adversarial behavior. Over the past decade, extensive research [12,16,29,42] has focused on combating *malicious* servers that may collude and return incorrect answers, potentially leading the client to reconstruct a wrong database entry.

Beimel and Stahl [6] introduced the notion of *b-Byzantine robust* k-out-of-ℓ PIR (BRPIR) [4,35,42], which allows a client to retrieve the correct value if any k out of the ℓ servers respond and at most b out of the k responding servers are malicious (called "Byzantines") and return incorrect answers. If $b = 0$, such schemes are simply called k-out-of-ℓ robust PIR (RPIR) schemes. They showed that any a-out-of-ℓ RPIR scheme can be used to construct a b-Byzantine robust k-out-of-ℓ PIR scheme for any $a < k < \ell$ and $b \leq (k - a)/2$. This class of BRPIR schemes can handle the cases where a minority of the responding servers (i.e., $b < k/2$) are malicious and provide incorrect responses. However, they require the client to execute a reconstruction algorithm whose running time may be exponential in k, the number of responding servers. For the same setting of $b < k/2$, Kurosawa [29] proposed an efficient BRPIR scheme with

polynomial time reconstruction, which is based on the RPIR scheme of Woodruff and Yekhanin [38] and utilizes the Berlekamp-Welch decoding algorithm [37] for Reed-Solomon codes.

While conventional BRPIR schemes are only applicable when $b < k/2$, they are incapable of addressing scenarios where a majority of the responses are erroneous. In practical situations, due to reasons such as (1) outdated databases, (2) poor or unstable mobile networks, and (3) significant communication channel noise, it is likely that most responses could be incorrect. Thus, it becomes imperative to develop a BRPIR scheme that can support higher error tolerance. The *list-decodable* BRPIR schemes of [12,16] represent a class of schemes that incorporate list-decoding techniques from coding theory into RPIR and enable one to handle the cases of $b \geq k/2$. In list-decodable BRPIR schemes, the client does not reconstruct a single definitive result but a list that contains the correct result. Goldberg [16] introduced the notion of list-decodable b-Byzantine-robust t-private k-out-of-ℓ PIR and constructed schemes that can handle b Byzantine servers for any $b < k - \sqrt{kt}$ (Table 1). The b-error correction achieved by [16] is particularly well-suited for Reed-Solomon code-based PIR schemes since it reaches the Johnson bound. The Johnson bound is a crucial threshold that delineates the upper limit of efficient polynomial-time list decoding for error-free recovery, ensuring that the maximum number of correctable errors in a polynomial-time framework does not exceed this limit. Reaching this bound indicates highly efficient and reliable error correction in managing erroneous responses in the RS code based PIR protocols. Devet, Goldberg, and Heninger [12] surpassed the Johnson bound with a *statistically correct* scheme that allows $b < k - t - 1$ (Table 1). However, the higher robustness is achieved at the price of statistical correctness, i.e., allowing a non-zero probability of failure in reconstruction. While the state-of-the-art PIR schemes [9,13] in the honest-but-curious server model achieve a sub-polynomial communication complexity, the list-decodable PIR schemes of [16,12] exhibit a *relatively high communication complexity* of $O(\ell n^{1/2})$. Neither [16] nor [12] gives a good estimation on the maximum size of the list that may be output by the client's reconstruction algorithm. In both works, the data block $x_i \in \mathbb{F}_p$ of interest is embedded into a degree-t polynomial and as the constant term. By Johnson Bound, the number of polynomials reconstructed by the client's algorithm is at most pk^2. However, the number of the candidate values for the data block (or equivalently the constant terms of the reconstructed polynomials) could be as high as p. In the worst-case, both schemes might output the entire field $\mathbb{F}_p$ as the list of the candidate values for x_i, rendering the schemes ineffective.

In most application scenarios of PIR, the database size n is significantly larger the parameters b, t, k, and ℓ. If we restrict to list-decodable BRPIR schemes that can handle a majority of malicious responding servers (i.e., $b \geq k/2$), the existing solutions have several limitations. First, they cannot achieve a communication complexity of $o(n^{1/2})$. This limitation persists even in scenarios where a minority of the responding servers are Byzantine (i.e., $b < k/2$) or when the privacy threshold t is relatively low (e.g., $t = O(1)$). Second, they lack a good estimation

on the size of the list that contains the block x_i of interest. In this paper, we are interested in list-decodable BRPIR schemes that can simultaneously handle a majority of malicious responding servers (i.e., $b \geq k/2$), achieve a communication complexity of $o(n^{1/2})$, and provide a nontrivial estimation on the list sizes.

Table 1. Comparisons for t-private list-decodable k-out-of-ℓ b-Byzantine-robust PIR schemes over database in $\mathbb{F}_p^n$

Reference	List-decodable	Byzantine Bound	Communication Complexity	Maximum List Size
[16]	Perfect	$k-\sqrt{kt}$	$O(\ell n^{1/2})$	p
[12]	Statistical	$k-t-1$	$O(\ell n^{1/2})$	p
Γ_1 (**Fig.** 2)	Perfect	$k-2$	$O(\ell w_1 n^{1/w_1})$ †	$(\frac{k}{k-b})^{w_1 t}$
Γ_2 (**Fig.** 3)	Perfect	$k-\sqrt{kt}$	$O(\ell w_2 n^{1/w_2})$ ‡	$2k$

† w_1 is a parameter chosen to balance the tradeoff between communication complexity and list size, satisfying $w_1 < \frac{2k-2b-1}{t}$.

‡ w_2 is a constant defined as $w_2 = \left\lfloor \frac{(k-b)^2}{kt} \right\rfloor$.

1.1 Our Results

In this paper, we construct two perfect L-list decodable b-Byzantine-robust t-private k-out-of-ℓ PIR schemes Γ_1 and Γ_2 (see **Table** 1). In both schemes, the Byzantine robustness parameter b can be at least $k/2$, the communication complexity is $o(n)$, and the list size L solely depends on the number k of responding servers and the Byzantine robustness b, and is independent of the size p of the space where each data block is taken from.

Byzantine robustness. The scheme Γ_1 has Byzantine robustness $b < k - \sqrt{kt}$, which is comparable to [16]. The scheme Γ_2 has a Byzantine robustness of $b \leq k-2$ and surpasses both the Byzantine robustness $b < k - \sqrt{kt}$ of [16] and the Byzantine robustness $b < k - t - 1$ of [12].

Communication complexity. Compared with [16] and [12], our schemes achieve a substantially lower communication complexity of $o(n^{1/2})$. The communication efficiency of our schemes is particularly high when the privacy threshold t is low. For instance, with parameters $(k, b, t) = (20, 12, 1)$, the communication complexity of the schemes Γ_1 and Γ_2 can be as low as $O(n^{1/14})$ and $O(n^{1/4})$, respectively. In contrast, for the same values of (k, b, t), the communication complexity of [16] and [12] is $O(n^{1/2})$. More detailed comparisons are provided in **Table.** 2.

List size. The scheme Γ_1 is L-list decodable for $L = (k/(k-b))^{w_1 t}$, where $w_1 < (2k-2b-1)/t$ is a parameter chosen by the client to balance the tradeoff between the communication complexity and the list size. The scheme Γ_2 is L-list decodable for $L = 2k$. Furthermore, when $b > k/2$ is large, the list size of Γ_2 can

be as small as $L = k$. Since the size of the finite field p in list-decodable BRPIR is much larger than k, b, t, compared with the p-list decodable schemes of [16] and [12], our schemes give substantial improvements.

1.2 Background

From RPIR to List-Decodable BRPIR. The starting point of our construction is the robust PIR (RPIR) scheme introduced by Woodruff and Yekhanin [38]. Their work established a powerful framework that reduces the problem of private information retrieval to the task of reconstructing a low-degree univariate polynomial. A detailed description of the Woodruff-Yekhanin RPIR scheme is provided in Section 2.4. In Woodruff-Yekhanin RPIR scheme, the client obtains both the values and the derivatives of a polynomial $f(\lambda)$ at k positions, that is, $\{f(\lambda_j), f'(\lambda_j)\}_{j\in[k]}$. The degree of this polynomial is set to be at most $2k-1$, so that it can be uniquely reconstructed from these evaluations and derivatives. Moreover, the higher the degree of $f(\lambda)$, the lower the communication complexity of the scheme, which leads to improved efficiency. A related construction by Beimel and Stahl [6] addresses the setting with Byzantine servers. When the number of corrupted servers is at most b, and at least $k-b$ evaluations and derivatives are correct, their scheme reconstructs a polynomial of degree at most $2(k-2b)-1$ at each of the $k-2b$ positions. A voting-like mechanism is then employed across these reconstructions to recover the correct polynomial $f(\lambda)$. However, when $b \geq k/2$, unique decoding strategies such as voting are no longer viable. In this case, one must resort to list decoding.

Sudan list decoding algorithm. When the number of Byzantine servers exceeds the threshold for unique decoding, list decoding becomes necessary. Sudan [34] proposed one of the earliest list decoding algorithms for Reed–Solomon codes, which can be viewed as a special case of univariate multiplicity codes. Given k pairs (λ_j, α_j), the algorithm outputs all low-degree polynomials $\tilde{f}(\lambda)$ such that $\tilde{f}(\lambda_j) = \alpha_j$ for at least $k-b$ indices $j \in [k]$, where b is the number of errors. A detailed description of the Sudan algorithm is provided in Section 2.3.

1.3 Our Approach

Building on the polynomial reconstruction framework by Woodruff and Yekhanin in [38], we develop two perfect list-decodable PIR schemes that remain effective even when the fraction of malicious servers exceeds the unique decoding threshold. The term "perfect" indicates that the list produced by this scheme always contains the queried result. Our goal is to retain the high-degree structure of the polynomial $f(\lambda)$, thereby minimizing communication complexity, while extending the scheme's robustness through list decoding techniques.

Recall that when the number of Byzantine servers b is less than $k/2$, unique decoding remains feasible, and mechanisms like majority voting over polynomial reconstructions at different positions can correctly recover the polynomial $f(\lambda)$.

However, once $b \geq k/2$, unique decoding breaks down, and we must resort to list decoding—outputting a small list of candidate polynomials that are guaranteed to contain the correct one.

The challenge in this setting is twofold: first, to design a decoding algorithm that produces only a constant-size list (or only polynomial-size in k), and second, to keep the degree of $f(\lambda)$ as high as possible, so as to preserve the communication efficiency inherited from the Woodruff-Yekhanin RPIR structure. To address the challenge, we introduce two distinct decoding strategies in our constructions, each leading to a perfect list-decodable PIR scheme.

In the first scheme, Γ_1, we introduce a method called *overinterpolation*. This method imposes a strict upper bound on the degree of any admissible interpolated polynomial, allowing us to discard inconsistent candidates while ensuring that the correct polynomial remains in the list. For example, we interpolate a polynomial from each subset of $k-b-1$ out of k values and retain only those polynomials whose degree is at most $(k-b)/2$. By carefully tuning this degree threshold, we can effectively bound the list size within a polynomial in k. This method is conceptually simple and yields both a small list size and low computational complexity when $\binom{k}{b}$ is small. However, when $\binom{k}{b}$ becomes large—if b and $k-b$ are both in $O(k)$ —the computational cost grows exponentially with k. To address such a situation, we further develop an optimized algorithm that reduces the overall complexity to a polynomial in k, even in such parameter settings.

In the second scheme, Γ_2, we propose a new list-decoding algorithm for order-1 multiplicity codes that follows Sudan list decoding algorithm. Unlike Sudan list decoding algorithm, which operates on Reed–Solomon codewords consisting of the point-value pair $(\lambda, f(\lambda))$, our algorithm works with multiplicity codewords composed of the point-value-derivative tuple $(\lambda, f(\lambda), f'(\lambda))$. Given k tuples $(\lambda_j, \alpha_j, \beta_j)$, our list decoding algorithm constructs a pair of polynomials: a base interpolating polynomial $Q^{\text{base}}(\lambda, \alpha)$, and an extended polynomial $Q^{\text{ext}}(\lambda, \alpha, \beta)$ that encodes additional derivative information. Following a process analogous to Sudan's approach, we identify a list of candidate polynomials $\tilde{f}(\lambda)$ that are consistent with at least $k-b$ of the given evaluations and derivatives. A key requirement of this approach is that the degree of the target polynomial $f(\lambda)$ must not exceed $(k-b)^2/k$. Compared to the Woodruff-Yekhanin RPIR scheme, where the degree reaches $2k-1$, this constraint results in a higher communication overhead. Nonetheless, it enables decoding in the environments where unique decoding is no longer feasible, while ensuring that both the list size and decoding time remain polynomial in k.

Both constructions can be seen as perfect list-decodable generalizations of the Woodruff-Yekhanin scheme, and are particularly suited for the environments in which unique decoding is provably impossible.

1.4 Related Work

The investigation of Byzantine robust PIR (BRPIR) schemes and list-decodable BRPIR schemes has made significant strides in addressing the challenges posed by malicious servers in PIR scenarios.

Byzantine Robust PIR (BRPIR) schemes[4,6,29,35,42] enable the client to both correctly retrieve the desired database element despite the presence of a limited number of malicious servers but also allow the client to identify which servers are acting maliciously. Beimel first introduced the idea of Byzantine robustness in [6]. Given an a-out-of-ℓ robust PIR scheme, they provided a general construction from an a-out-of-ℓ robust PIR scheme to a k-out-of-ℓ, b-Byzantine-robust PIR scheme for any $a < k < \ell$ with $b = (k-a)/2$. Applying this construction to the robust PIR scheme proposed by Woodruff and Yekhanin [38], which retrieves hidden data blocks by interpolating polynomials, one can obtain an efficient t-private k-out-of-ℓ b-Byzantine-robust PIR scheme, where $b < k/2$, and the communication complexity of this scheme is $O(n^{1/\lfloor(2(k-2b)-1)/t\rfloor})$. However, this scheme requires the client to perform computations that grow exponentially with the number of responding servers k, leading to high computational complexity for the client. By extending the Berlekamp-Welch algorithm [37], a decoding algorithm for Reed-Solomon (RS) codes, to the case of first-order derivatives, Kurosawa [29] reduced the client-side computation of the construction by Beimel and Stahl to polynomial levels. Nonetheless, the Byzantine tolerance remains bounded by $b < k/2$.

List-Decodable BRPIR schemes [12,16] are suitable for scenarios where conventional BRPIR schemes fail to operate, particularly when $b \geq k/2$. This idea of list-decoding in BRPIR was first introduced by Goldberg [16] to improve the bound of byzantine tolerance b. Goldberg proposed a t-private list-decodable k-out-of-ℓ b-Byzantine-robust PIR scheme for any $b < k - \sqrt{kt}$. In Goldberg's scheme, the database is structured as a matrix. The client retrieves a row of this matrix from each server in the form of a degree-t RS codeword. The data block is then reconstructed using the Guruswami-Sudan list decoding algorithm [20], which is a list decoding algorithm for RS codes, applied to the codewords of RS codes. For cases where $t < k/4$, this scheme can achieve $b \geq k/2$. Devet, Goldberg and Heninger [12] improved the Byzantine bound of list-decodable BRPIR by proposing a *statistical* list-decodable BRPIR scheme with byzantine bound $b < k - t - 1$. In both works of [16] and [12], the client's desired data block $x_i \in \mathbb{F}_p$ is embedded in the constant term of a degree-t polynomial, which is then subjected to list decoding. According to the bound given by the Johnson Bound, the number of decoded polynomials is at most pk^2. However, the possible values for the data block are at most p. In the worst-case scenario, both schemes might output the entire $\mathbb{F}_p$ as the candidate list for x_i, rendering the schemes ineffective.

There are other works addressing malicious servers that focus solely on error detection without aiming to retrieve the correct result. Some of these approaches achieve higher efficiency and greater Byzantine server tolerance compared to BRPIR.

Verifiable PIR (VPIR) schemes in both the multi-server model [41] and the single-server model [43], ensures that the client can identify the byzantine servers, that is, can tell which server is malicious. However, it does not guarantee the

recovery of the correct element, or even a small list of potential candidates. The security of VPIR is weaker than that of BRPIR. However, this allows VPIR to accommodate a significantly higher number of malicious servers and greatly reduces communication complexity.

Error Detecting PIR (EDPIR) [8,11,15,23,24,28,44] schemes allow the client to detect the existence of incorrect answers provided by malicious servers, though they do not guarantee identifying which servers are malicious or reconstructing the correct value.

In our work, we devise a list decoding method for a variant of Reed-Solomon (RS) codes that consider first-order derivatives. By integrating this list decoding method into the Woodruff-Yekhanin robust PIR scheme, we obtain a list-decodable BRPIR scheme. This variant of RS codes, which incorporates first-order derivatives, can be viewed as a special case of univariate multiplicity codes.

Univariate Multiplicity Codes [19] are variants of Reed–Solomon (RS) codes that are constructed by evaluating a polynomial along with all of its derivatives up to order s. Notably, RS codes correspond to the special case when $s = 0$. In recent years, there has been significant progress in the list decoding of univariate multiplicity codes [17,21,22,26,27], with the decoding radius—corresponding to the Byzantine tolerance b/k in our framework—reaching up to $1 - R - \epsilon$ for any $\epsilon > 0$, where R is the code rate. This matches the list decoding capacity $1 - R$. However, when restricted to only first-order derivatives and function values, i.e., order-1 multiplicity codes, these results no longer apply. We provide a detailed discussion of these limitations in Section 2.3.

2 Preliminaries

2.1 Notation

We use bold lower-case letters to denote vectors. For any vector $\mathbf{v}$ (resp. $\mathbf{v}_j$) of length m and any $c \in [m]$, we denote by v_c (resp. $v_{j,c}$) the c-th element of $\mathbf{v}$ (resp. $\mathbf{v}_j$), meaning that $\mathbf{v} = (v_1, \ldots, v_m)$ (resp. $\mathbf{v}_j = (v_{j,1}, \ldots, v_{j,m})$). For any finite set A, we denote by $|A|$ the *cardinality* of A. For any integer $n > 0$, we denote $[n] = \{1, 2, \ldots, n\}$ and $\{a_j\}_{j\in[n]} = \{a_1, a_2, \ldots, a_n\}$. For any two vectors $\mathbf{u}, \mathbf{v}$ of the same length, we denote by $\langle \mathbf{u}, \mathbf{v} \rangle$ the *inner product* of $\mathbf{u}$ and $\mathbf{v}$. For any prime p, we denote by $\mathbb{F}_p$ the *finite field* of p elements and denote by $\mathbb{F}_p^n$ the set of all length-n vectors over $\mathbb{F}_p$. For any polynomial $f(\lambda) \in \mathbb{F}_p[\lambda]$ and any integer $s \geq 0$, we denote by $f^{(s)}(\lambda)$ the *order-s derivative* of f with respect to λ and denote by $f^{(\leq s)}(\lambda) = (f^{(0)}(\lambda), f^{(1)}(\lambda), ..., f^{(s)}(\lambda))$ the *order-s evaluation* of f at λ. In particular, $f^{(0)}(\lambda) = f(\lambda)$ and $f^{(1)}(\lambda) = f'(\lambda)$ are the evaluation and order-1 derivative of f at λ, respectively.

We will use the following variables throughout the paper:

- ℓ: the total number of servers.
- k: the number of responding servers.

- t: the number of servers that may collude to learn the retrieval index.
- b: the number of servers that may collude to respond incorrectly.
- n: the database size.
- p: the database block size, each entry in the database is an element in $\mathbb{F}_p$.
- $\mathbf{x} = (x_1, \ldots, x_n)$: the database, which is a vector in $\mathbb{F}_p^n$.
- w: a degree parameter.
- m: the least positive integer such that $\binom{m}{w} \geq n$.
- C: a subset of $\mathbb{F}_p^m$ and a constant weight code of length m and weight w.
- E: a *public* 1-to-1 encoding function.
- F: an encoding of the database $\mathbf{x}$ such that $F(E(i)) = x_i$.

2.2 Constant-weight Code

A *code* C of length n over $\mathbb{F}_p$ is a subset of $\mathbb{F}_p^n$. For any two *codewords* $\mathbf{u}, \mathbf{v} \in C$, the *Hamming distance* between $\mathbf{u}, \mathbf{v}$ is the number of coordinates where $\mathbf{u}, \mathbf{v}$ differ and denoted by $d_H(\mathbf{u}, \mathbf{v}) = |\{i \in [n] : \mathbf{u}_i \neq \mathbf{v}_i\}|$. The *minimum distance* of a code C is the least Hamming distance between any two different codewords in C and denoted by $d_H(C) = \min_{\mathbf{u},\mathbf{v} \in C, \mathbf{u} \neq \mathbf{v}} d_H(\mathbf{u}, \mathbf{v})$. For any codeword $\mathbf{u} \in C$, the Hamming weight of $\mathbf{u}$ is the number nonzero coordinates of $\mathbf{u}$ and denoted by $\mathsf{wt}(\mathbf{u}) = d_H(\mathbf{u}, \mathbf{0})$.

For any integers $m, d, w > 0$, an (m, d, w) *constant-weight code* [7] is a binary code of length m, minimum Hamming distance d, and Hamming weight w. The *maximum* size of an (m, d, w) constant weight code is denoted by $A(m, d, w)$. For any integers $m, \delta, w > 0$, Agrell, Vardy and Zeger[1] showed an upper bound on $A(m, 2\delta, w)$.

Theorem 1. (Agrell, Vardy and Zeger [1], Theorem 12)

$$A(m, 2\delta, w) \leq \frac{\binom{m}{w-\delta+1}}{\binom{w}{w-\delta+1}}.$$

2.3 Univariate Multiplicity Codes and their List-decoding Algorithms

For any integer $s \geq 0$, an *order-s univariate multiplicity code* C of length k for degree-w polynomials over $\mathbb{F}_p$ is an error-correcting code that encodes each polynomial $f(\lambda) \in \mathbb{F}_p[\lambda]$ with $\deg f \leq w$ into the codeword

$$C_f = \left(f^{(\leq s)}(\lambda_j)\right)_{j=1}^{k},$$

where $\lambda_1, \ldots, \lambda_k$ are pairwise distinct elements of $\mathbb{F}_p$. In particular, the renowned Reed–Solomon codes arise when $s = 0$. In this work, we specialize to $s = 1$.

Given a set of k tuples $\{(\lambda_j, \alpha_{0,j}, \ldots, \alpha_{s,j})\}_{j=1}^{k}$, a *list decoding* algorithm that corrects b errors for a univariate multiplicity code C may identify all polynomials $\tilde{f}(\lambda)$ of degree $\leq w$ such that $\tilde{f}^{(\leq s)}(\lambda_j) = (\alpha_{0,j}, \ldots, \alpha_{s,j})$ for at least $k-b$ distinct

indices $j \in [k]$ and outputs the list of all such polynomials. If the size of the list is at most L, then the code is called *L-list decodable*.

For Reed-Solomon codes (i.e., $s = 0$), Sudan [34] has a list decoding algorithm that interpolates a bivariate polynomial

$$Q(\lambda, \alpha) = \sum_{c=0}^{D/w} Q_c(\lambda)\alpha^c \tag{1}$$

of $(1, w)$-weighted degree $\leq D$ by solving k constraints of the form

$$Q(\lambda_j, \alpha_{0,j}) = 0, j \in [k],$$

and then outputs a list of candidate polynomials $\tilde{f}(\lambda)$ such that $(\alpha - \tilde{f}(\lambda))|Q(\lambda, \alpha)$, where D is a carefully chosen parameter and $\deg(Q_c(\lambda)) \leq D - cw$ for all $0 \leq c \leq D/w$.

Recent list decoding algorithms [17,20,21,22,26,27] for the general order-s univariate multiplicity codes extended the basic idea [34] of *interpolating* a multivariate polynomial Q and then *factoring* Q to find all candidate polynomials $\tilde{f}(\lambda)$. An early list-decoding algorithm of univariate multiplicity codes given by Guruswami and Wang [21,22] considered

$$Q(\lambda, \alpha_0, \ldots, \alpha_r) = P(\lambda) + \sum_{c=0}^{r} Q_c(\lambda)\alpha_c, \tag{2}$$

an $(r + 2)$-variate polynomial of degree $\leq D$, where $r \leq s$ and D are carefully chosen parameters, $P(\lambda)$ is of degree $\leq D$, and $Q_c(\lambda)$ is of degree $\leq D - w + 1$ for all $c \in \{0, \ldots, r\}$. In particular, the polynomial Q may be viewed as a function of λ if $\alpha_0, \ldots, \alpha_s$ are all viewed as functions of λ. They constructed $s - r$ new polynomials $\{\mathcal{D}^c Q(\cdot)\}_{c=1,\ldots,s-r}$ from Q by successively applying a special operator $\mathcal{D}$ that essentially differentiates Q with respect to λ using the well-known chain rule but requires that $\mathcal{D}(\alpha_c) = \alpha_{c+1}$ (instead of $\mathcal{D}(\alpha_c) = \alpha_c'$) for all $c = 0, 1, \ldots, s - 1$. They developed $(s - r + 1)k$ constraints that

$$Q(\lambda_j, \alpha_{0,j}, \ldots, \alpha_{r,j}) = 0 \text{ and } \mathcal{D}^c Q(\lambda_j, \alpha_{0,j}, \ldots, \alpha_{s,j}) = 0, \ c \in [s - r], j \in [k].$$

In their construction, the parameter D is prescribed as

$$D = \left\lfloor \frac{k(s - r + 1) - w}{r + 1} \right\rfloor,$$

which provides resilience against up to $b = k - \left\lfloor \frac{D+w}{s-r+1} \right\rfloor$ errors. The advantage of this construction lies in its ability to achieve a trade-off between computational complexity and error tolerance by appropriately choosing the parameters r and D. However, in the special case $s = 1$, which is crucial for our construction of list-decodable BRPIR schemes, it can correct at most $b < k/2$ errors, regardless of the choice of $r = 0$ or $r = 1$.

The recent works in [17,27] establish that order-s univariate multiplicity codes of length k, encoding degree-w polynomials over $\mathbb{F}_p$, are L-list decodable from up to $b = \left(1 - \frac{w}{sk} - \epsilon\right) k$ errors, where $L = \left(\frac{1}{\epsilon}\right)^{O\left(\frac{1}{\epsilon}\log\frac{1}{\epsilon}\right)}$ and $\epsilon \geq \sqrt{16/s}$. When s is large, the list decoding error tolerance b/k of this scheme approaches the capacity $1 - R$, where $R = w/sk$ is the code rate. However, for small values of s, such as $s = 1$, the constraint $\epsilon > 1$ renders this result inapplicable.

2.4 Woodruff-Yekhanin RPIR Scheme

$\underline{(\{q_j\}_{j\in[\ell]}, \mathsf{aux}) \leftarrow \Gamma.\mathcal{Q}(n, i)}$:

1. Randomly choose t vectors $\mathbf{r}_1, ..., \mathbf{r}_t \in \mathbb{F}_p^m$ and set $G(\lambda) = E(i) + \sum_{s=1}^{t} \lambda^s \mathbf{r}_s$.
2. For each $j \in [\ell]$, set $q_j = G(\lambda_j)$. Set $\mathsf{aux} = (\{\lambda_j\}_{j\in[\ell]}, \{\mathbf{r}_s\}_{s\in[t]})$.
3. Output $(\{q_j\}_{j\in[\ell]}, \mathsf{aux})$.

$\underline{a_j \leftarrow \Gamma.\mathcal{A}(\mathbf{x}, q_j)}$:

1. Compute $u_j = F(q_j)$. For each $c \in [m]$, set $v_{j,c} = \left.\dfrac{\partial F(\mathbf{z})}{\partial z_c}\right|_{q_j}$.
2. Let $\mathbf{v}_j = (v_{j,1}, ..., v_{j,m})$. Output $a_j = (u_j, \mathbf{v}_j)$.

$\underline{x_i \leftarrow \Gamma.\mathcal{R}(i, \{a_j\}_{j\in[\ell]}, \mathsf{aux})}$:

1. Parse $\mathsf{aux} = (\{\lambda_j\}_{j\in[\ell]}, \{\mathbf{r}_s\}_{s\in[t]})$. Construct the polynomial $G(\lambda) = E(i) + \sum_{s=1}^{t} \lambda^s \mathbf{r}_s$.
2. Wlog, suppose the first k servers respond and denote $a_j = (u_j, \mathbf{v}_j)$ for all $j \in [k]$. For each $j \in [k]$, let $\alpha_j = u_j$, $\beta_j = \langle \mathbf{v}_j, G'(\lambda_j)\rangle$.
3. Interpolate a polynomial $f(\lambda)$ of degree at most $2k-1$ such that $f(\lambda_j) = \alpha_j$ and $f'(\lambda_j) = \beta_j$ for all $j \in [k]$.
4. Output $x_i = f(0)$.

Fig. 1. Woodruff-Yekhanin (t, k, ℓ)-RPIR scheme Γ.

Woodruff and Yekhanin [38] constructed a t-private k-out-of-ℓ RPIR scheme, referred to as a (t, k, ℓ)-RPIR scheme, Γ (see **Fig.** 1) with communication complexity $\mathcal{O}(\frac{k\ell}{t} \log \ell \cdot n^{1/\lfloor (2k-1)/t \rfloor})$, where n is the size of the database $\mathbf{x} = (x_1, \ldots, x_n)$.

For a degree parameter w defined as

$$w = \lfloor (2k-1)/t \rfloor, \tag{3}$$

their scheme chooses an integer $m = O(wn^{1/w})$ such that $\binom{m}{w} \geq n$, encodes the indices of the n database elements as binary vectors of length m and weight w with a *public* 1-to-1 function

$$E : [n] \to \{0, 1\}^m,$$

and represents the database $\mathbf{x}$ as

$$F(\mathbf{z}) = F(z_1, \ldots, z_m) = \sum_{j=1}^{n} x_j \cdot \prod_{c:E(j)_c=1} z_c, \tag{4}$$

a homogeneous m-variate polynomial of degree w that satisfies

$$F(E(i)) = x_i \tag{5}$$

for all $i \in [n]$. The client reduces the problem of privately retrieving x_i from ℓ servers to the problem of privately evaluating $F(E(i))$ with the ℓ servers. To this end, a prime $p > \ell$ is chosen and F is interpreted as a polynomial over the finite field $\mathbb{F}_p$. For each $j \in [\ell]$, the j-th server is associated with a nonzero field elements λ_j. In particular, the field elements $\{\lambda_j\}_{j\in[k]}$ are distinct and can be made public. To retrieve x_i, the client chooses t vectors $\mathbf{r}_1, \mathbf{r}_2, \ldots, \mathbf{r}_t \in \mathbb{F}_p^m$ uniformly at random and generates a vector

$$G(\lambda) = E(i) + \sum_{s=1}^{t} \lambda^s \mathbf{r}_s \tag{6}$$

of m polynomials of degree t. It sends a query

$$q_j = G(\lambda_j)$$

to the j-th server for all $j \in [\ell]$ and keeps the information $\mathsf{aux} = (\{\lambda_j\}_{j\in[\ell]}, \{\mathbf{r}_s\}_{s\in[t]})$ for later use. The j-th server is expected to compute $u_j = F(q_j)$ and $\mathbf{v}_j = (v_{j,1}, \ldots, v_{j,m})$, where

$$v_{j,c} = \left.\frac{\partial F(\mathbf{z})}{\partial z_c}\right|_{q_j}$$

for all $c \in [m]$, and reply with

$$a_j = (u_j, \mathbf{v}_j).$$

Wlog, suppose the first k servers respond, the client interpolates the univariate polynomial

$$f(\lambda) = F(G(\lambda)) \tag{7}$$

of degree $wt \le 2k - 1$ from the $2k$ values

$$f(\lambda_j) = u_j, \quad f'(\lambda_j) = \langle \mathbf{v}_j, G'(\lambda_j) \rangle, \quad j \in [k]$$

and outputs $f(0)$, which is equal to x_i as

$$x_i = F(E(i)) = F(G(0)) = f(0).$$

The scheme is t-private such that any t colluding servers learn no information about i (or equivalently $E(i)$), because every element of $E(i)$ is secret-shared

among the servers with Shamir's t-private threshold secret sharing scheme [33], using the vector $G(\lambda)$ of m random polynomials.

Note that the client sends a length-m vector in $\mathbb{F}_p$ to each server and each server returns a length-$(m+1)$ vector in $\mathbb{F}_p$. Given that $m = O(wn^{1/w})$, if the prime p is chosen such that $\ell < p \leq 2\ell$, then the communication complexity of this scheme is $(2m+1)\ell \log p = O(\frac{k\ell \log \ell}{t} n^{1/\lfloor \frac{2k-1}{t} \rfloor})$.

In the default configuration of [38], λ_j is set to j for all $j \in [\ell]$ and eliminates the necessity to include $\{\lambda_j\}_{j=1}^{\ell}$ in aux.

3 List-Decodable BRPIR model

In this section, we formally define a model for list-decodable BRPIR, which generalizes the standard BRPIR of [6] by allowing the reconstructing algorithm to output a list that contains the data item being retrieved.

Similar to the standard BRPIR, the list-decodable BRPIR allows a client to retrieve a data item from a database replicated among multiple servers, even if some of the servers are silent or malicious, where the silent servers simply fail to respond and the malicious servers respond incorrectly. While the standard BRPIR may enable correct retrieval when a minority of the responding servers are malicious, what really differentiates our list decodable BRPIR from the standard BRPIR is its ability to handle the much trickier case that a majority of the responding servers are malicious and thus offer stronger robustness for more adversarial environments.

Informally, an L-list-decodable b-Byzantine-robust t-private k-out-of-ℓ PIR scheme is a protocol between ℓ servers $\{\mathcal{S}_j\}_{j\in[\ell]}$, each storing a copy of the same database $\mathbf{x} = (x_1, \ldots, x_n) \in \mathbb{F}_p^n$, and a client $\mathcal{C}$ that is interested in a block x_i of the database. It allows the client to output a list of size $\leq L$ that contains x_i, as long as at least k out of the ℓ servers respond and at most b out of the responding servers provide incorrect answers; and

Definition 1 (List-decodable BRPIR). *An* L-list-decodable b-Byzantine-robust t-private k-out-of-ℓ PIR *scheme* $\Gamma = (\mathcal{Q}, \mathcal{A}, \mathcal{R})$ *for a client* $\mathcal{C}$ *and* ℓ *servers* $\mathcal{S}_1, \ldots, \mathcal{S}_\ell$ *consists of three algorithms that can be described as follows:*

- $(\{q_j\}_{j\in[\ell]}, \mathsf{aux}) \leftarrow \mathcal{Q}(n, i)$*: This is a randomized* querying algorithm *for the client* $\mathcal{C}$*. It takes the database size* n *and a retrieval index* $i \in [n]$ *as input, and outputs* ℓ *queries* $\{q_j\}_{j\in[\ell]}$*, along with an auxiliary information* aux*. For each* $j \in [\ell]$*, the query* q_j *will be sent to the server* $\mathcal{S}_j$*. The auxiliary information* aux *will be used later by the client for reconstruction.*
- $a_j \leftarrow \mathcal{A}(\mathbf{x}, q_j)$*: This is a deterministic* answering algorithm *for the server* $\mathcal{S}_j$ $(j \in [\ell])$*. It takes a database* $\mathbf{x} = (x_1, \ldots, x_n)$ *and the query* q_j *as input and outputs an answer* a_j*.*
- $\mathsf{output\_list} \leftarrow \mathcal{R}(i, \{a_j\}_{j\in[\ell]}, \mathsf{aux})$*: This is a deterministic* reconstructing algorithm *for the client* $\mathcal{C}$*. It uses the retrieval index* i*, the answers* $\{a_j\}_{j\in[\ell]}$ *and the auxiliary information* aux *to reconstruct* x_i*, where* a_j *is set to* $\perp$ *if a server* $\mathcal{S}_j$ *does not respond. The output is a list of size at most* $L(= k^{O(1)})$ *that contains the queried data block* x_i*.*

Correctness. *Informally, the scheme Γ is considered* correct *if the* output_list *generated by the reconstructing algorithm $\mathcal{R}$ is always of size $\leq L$ and includes the target block x_i, provided that at least k out of the ℓ servers respond and at most b of the responses are incorrect. Formally, the scheme Γ is* correct *if for any n, any $\mathbf{x} \in \mathbb{F}_p^n$, any $(\{q_j\}_{j\in[\ell]}, \mathsf{aux}) \leftarrow \mathcal{Q}(n, i)$, any answers $\{a_j\}_{j\in[\ell]}$ such that*

$$|\{j \in [\ell] : a_j \neq \perp\}| \geq k,$$
$$|\{j \in [\ell] : a_j \notin \{\perp, \mathcal{A}(\mathbf{x}, q_j)\}| \leq b,$$

and any output_list $\leftarrow \mathcal{R}(i, \{a_j\}_{j\in[\ell]}, \mathsf{aux})$*, it holds that*

$$\Pr\left[(|\mathsf{output\_list}| \leq L) \wedge (x_i \in \mathsf{output\_list})\right] = 1. \tag{8}$$

***t*-Privacy**. *Informally, the scheme Γ is considered* t-private *if any collusion of $\leq t$ servers learns no information about the client's retrieval index i. Formally, the scheme Γ is* t-private *if for any n, any $i_1, i_2 \in [n]$, and any set $T \subseteq [\ell]$ of size $\leq t$, the distributions of $\mathcal{Q}_T(n, i_1)$ and $\mathcal{Q}_T(n, i_2)$ are identical, where $\mathcal{Q}_T$ denotes the concatenation of the j-th output of $\mathcal{Q}$ for all $j \in T$, i.e., $\{q_j\}_{j\in T}$*

Remark 1. Our correctness property requires the scheme Γ to satisfy Eq. (8). We call this kind of correctness *perfect*. By contrast, several existing BRPIR schemes [12] may satisfy a relaxed correctness property that guarantees

$$\Pr\left[(|\mathsf{output\_list}| \leq L) \wedge (x_i \in \mathsf{output\_list})\right] \geq 1 - \epsilon \tag{9}$$

for a very small number ϵ. When the failure probability ϵ is sufficiently small (e.g., negligible in the number of servers), the scheme is statistically reliable and we call this kind of correctness *statistical*.

Remark 2. For the ease of exposition, hereafter we denote any L-list-decodable b-Byzantine-robust t-private k-out-of-ℓ PIR scheme by (L, b, t, k, ℓ)-ldBRPIR. By default, when $t = 1$ we simply denote any $(L, b, 1, k, \ell)$-ldBRPIR as (L, b, k, ℓ)-ldBRPIR and refer to the property of 1-privacy as privacy.

The efficiency of an (L, b, t, k, ℓ)-ldBRPIR scheme is mainly measured by its communication complexity, which is the average number of bits that have to be communicated between the client and all servers, in order to retrieve one bit from the database.

Definition 2 (Communication Complexity). The *communication complexity* of the scheme Γ, denoted by $\mathsf{CC}_\Gamma(n)$, is defined as the average number of bits communicated per bit retrieved between the client and all servers, maximize over the choices of both the database $\mathbf{x} \in \mathbb{F}_p^n$ and the retrieval index $i \in [n]$, i.e.,

$$\mathsf{CC}_\Gamma(n) = \max_{\mathbf{x},i} \left(\frac{1}{\lceil \log_2 p \rceil} \sum_{j=1}^{\ell} (|q_j| + |a_j|) \right).$$

4 Perfect ldBRPIR based on Overinterpolation

In this section, we construct a perfectly correct (L, b, t, k, ℓ)-ldBRPIR scheme Γ_1 with list size $L = k^{O(1)}$ and Byzantine robustness $b \leq k-2$. The proposed scheme is most suitable for a small number of servers and comparably as efficient as a t-private $(k-b)$-out-of-ℓ RPIR scheme.

$\mathsf{output\_list} \leftarrow \Gamma_1.\mathcal{R}(i, \{a_j\}_{j\in[\ell]}, \mathsf{aux})$:

1. Parse $\mathsf{aux} = (\{\lambda_j\}_{j\in[\ell]}, \{\mathbf{r}_s\}_{s\in[t]})$. Construct the polynomial $G(\lambda) = E(i) + \sum_{s=1}^{t} \lambda^s \mathbf{r}_s$.
2. Wlog, suppose the first k servers respond and denote $a_j = (u_j, \mathbf{v}_j)$ for all $j \in [k]$. For each $j \in [k]$, let $\alpha_j = u_j, \beta_j = \langle \mathbf{v}_j, G'(\lambda_j)\rangle$.
3. Initialize a set $\mathsf{cp} = \emptyset$ to store the candidate polynomials. For any set $H \subseteq [k]$ of cardinality $k-b$,
   (3.1) Interpolate a polynomial $f_H(\lambda)$ such that for each $j \in H$, $f_H(\lambda_j) = \alpha_j, f'_H(\lambda_j) = \beta_j$.
   (3.2) If $f_H(\lambda)$ is of degree at most wt, add $f_H(\lambda)$ to the set cp.
4. Output $\mathsf{output\_list} = \{f_H(0) : f_H(\lambda) \in \mathsf{cp}\}$.

Fig. 2. Reconstructing algorithm $\Gamma_1.\mathcal{R}$ of the (L, b, t, k, ℓ)-ldBRPIR scheme Γ_1.

4.1 The Construction

The scheme Γ_1 retains the querying and answering algorithms of the Woodruff-Yekhanin RPIR scheme Γ, $(\Gamma_1.\mathcal{Q}, \Gamma_1.\mathcal{A}) = (\Gamma.\mathcal{Q}, \Gamma.\mathcal{A})$. The key differences between Γ_1 and Γ are as follows: **(I)** The degree parameter w is required to satisfy Eq. (3) in Γ, whereas in Γ_1 it is required to satisfy Eq. (11). **(II)** The reconstructing algorithm $\Gamma.\mathcal{R}$ in Γ is replaced by the reconstructing algorithm $\Gamma_1.\mathcal{R}$ in Γ_1, as shown in **Fig** 2.

The scheme Γ_1 achieves the expected list size and Byzantine robustness by invoking a novel reconstructing algorithm, which interpolates the polynomial $f(\lambda) = F(G(\lambda))$ of Eq. (7) with more server responses than necessary. More precisely, upon k out of ℓ servers responding, the client interpolates $f(\lambda)$ of degree-$wt(< 2(k-b)-1)$ with any $k-b$ of the responses, determines whether the interpolated polynomial is a possible candidate of $f(\lambda)$, and includes it into $\mathsf{output\_list}$ when it is indeed a possible candidate. As the crux of this idea, we require

$$\deg(f(\lambda)) < 2(k-b) - 1 \tag{10}$$

such that the $k-b$ evaluations and $k-b$ derivatives deduced from the $k-b$ server responses are more than necessary and thus enable us to determine whether each interpolated polynomial is indeed a possible candidate with its degree.

We refer to this technique of using more-than-necessary points to do polynomial interpolation as *overinterpolation*. In particular, for given k and b, the inequality (10) is met by choosing a weight parameter $w = O(1)$ in Woodruff-Yekhanin RPIR scheme (see Section 2.4) such that

$$w < \left\lfloor \frac{2(k-b)-1}{t} \right\rfloor. \tag{11}$$

The main observation about our overinterpolation technique includes: (1) whenever some of the $k-b$ server responses under consideration are incorrect, the interpolated polynomial will be of degree $> \deg(f(\lambda))$ with very large probability and thus be ruled out; and (2) whenever the server responses are all correct, the interpolated polynomial will be of degree $\deg(f(\lambda))$ with probability 1 and thus appear in output_list. In other words, the degree of $f(\lambda)$ serves as a *limit* and gives a filtering process that effectively eliminates numerous erroneous results while preserving the correct one. By adjusting this limit, we can regulate the number of polynomials that pass the filtering process. When the limit is set to an optimal value, the number of polynomials that pass the filtering process will be $k^{O(1)}$, thereby achieving the expected list size for any admissible Byzantine robustness parameter b.

4.2 Analysis

In this section, we show that the proposed scheme Γ_1 is indeed an (L, b, t, k, ℓ)-ldBRPIR scheme for $L = k^{O(1)}$ and $b \leq k-2$.

Correctness. To show that the proposed scheme is correct, we start with a technical lemma that gives an upper bound on the number of the points where two distinct degree d polynomials have the same order-1 evaluations.

Lemma 1. *Suppose that $f_1(\lambda)$ and $f_2(\lambda)$ are two distinct polynomials of degree d over a finite field $\mathbb{F}_p$. Then there exist at most $\lfloor \frac{d}{2} \rfloor$ field elements λ_j such that $f_1(\lambda_j) = f_2(\lambda_j)$ and $f_1'(\lambda_j) = f_2'(\lambda_j)$.*

Proof. Assume for contradiction that there exist $\theta \geq \lfloor \frac{d}{2} \rfloor + 1$ distinct field elements $\lambda_1, \lambda_2, \ldots, \lambda_\theta$ such that $f_1(\lambda_j) = f_2(\lambda_j)$ and $f_1'(\lambda_j) = f_2'(\lambda_j)$ for all $j \in [\theta]$. Then the nonzero polynomial $g(\lambda) = f_1(\lambda) - f_2(\lambda)$ is of degree at most d and satisfies

$$\forall j \in [\theta], \quad g(\lambda_j) = 0 \ \text{ and } \ g'(\lambda_j) = 0.$$

It follows that

$$\forall j \in [\theta], \quad (\lambda - \lambda_j)^2 \mid g(\lambda),$$

Hence, the nonzero polynomial $g(\lambda)$ must be of degree $\geq 2\theta > d$, which gives a contradiction. □

Note that the candidate polynomials in the set cp generated by our reconstructing algorithm $\Gamma_1.\mathcal{R}$ all have degree $\leq 2(k-b)-1$. Lemma 1 shows that any two polynomials from cp cannot have the same order-1 evaluations at too

many field elements. In the language of multiplicity codes, the polynomials in cp must be distant from each other. The following theorem shows that by appropriately choosing the parameters w, b, t and k, the size L of $\mathsf{output\_list}$ in our reconstructing algorithm can be made as small as $k^{O(1)}$.

Theorem 2. *The scheme Γ_1 is correct with list size $L = k^{O(1)}$ when $w < \lfloor \frac{2(k-b)-1}{t} \rfloor$ and $wt = O(1)$ with respect to k.*

Proof. Let $\mathsf{output\_list}$ be the list output by $\Gamma_1.\mathcal{R}$. As per Eq. (8), it suffices to show that **(I)** $x_i \in \mathsf{output\_list}$*, i.e., the list output by $\Gamma_1.\mathcal{R}$ always contains the expected block x_i*; and **(II)** $|\mathsf{output\_list}| \leq k^{O(1)}$*, i.e., the size L of the list output by $\Gamma_1.\mathcal{R}$ is $k^{O(1)}$, under the proposed choices of the parameters w, b, t and k.*

(I) Referring to the description of $\Gamma_1.\mathcal{R}$ in Figure 2, wlog the first k servers respond. Consider the degree-wt polynomial

$$f(\lambda) = F(G(\lambda)) = F(E(i) + \sum_{s=1}^{t} \lambda^s \mathbf{r}_s).$$

For every $j \in [k]$, it is easy to see that

$$\begin{aligned} f(\lambda_j) &= F(G(\lambda_j)) = F(q_j), \\ f'(\lambda_j) &= \sum_{c=1}^{m} \left.\frac{\partial F(\mathbf{z})}{\partial z_c}\right|_{G(\lambda_j)} \cdot G'(\lambda_j)_c \\ &= \langle \mathbf{v}_j, G'(\lambda_j) \rangle, \end{aligned}$$

where $G'(\lambda_j)_c$ denotes the c-th entry of the vector $G'(\lambda_j)$. Let $a_j = (\alpha_j, \beta_j)$ be the response of the j-th server for every $j \in [k]$. If at most b of the responses $\{a_j\}_{j\in[k]}$ are incorrect (where "the response a_j is incorrect" means that either $\alpha_j \neq F(q_j)$ or $\beta_j \neq \langle \mathbf{v}_j, G'(\lambda_j) \rangle$), then for at least $k - b$ indices $j \in [k]$ we have

$$f^{(\leq 1)}(\lambda_j) = (\alpha_j, \beta_j).$$

When the set H in $\Gamma_1.\mathcal{R}$ is composed of the indices of $k - b$ correct responses, the interpolated polynomial $f_H(\lambda)$ is exactly equal to $f(\lambda)$ and thus has degree $\leq wt (< 2(k-b)-1)$ and results in $f_H(0) = f(0) = F(E(i)) = x_i$ being included into the set $\mathsf{output\_list}$.

(II) Referring to the description of $\Gamma_1.\mathcal{R}$ in **Fig.** 2, as $\mathsf{output\_list} = \{f_H(0) : f_H(\lambda) \in \mathsf{cp}\}$, it is trivial to see that

$$|\mathsf{output\_list}| \leq |\mathsf{cp}|, \tag{12}$$

where the equality occurs if and only if all polynomials in cp have different constant terms. Let

$$\mathcal{H} = \{H : H \subseteq [k], |H| = k - b, \deg(f_H(\lambda)) \leq w\}.$$

be the set of all subsets of $[k]$ of cardinality $k-b$ that gives an interpolated polynomial of degree $\leq w$. According to the construction of cp, it is trivial to see that $\mathsf{cp} = \{f_H(\lambda) : H \in \mathcal{H}\}$. Therefore,

$$|\mathsf{cp}| \leq |\mathcal{H}|. \tag{13}$$

It is possible that $|\mathsf{cp}| < |\mathcal{H}|$ because different subsets $H_1, H_2 \in \mathcal{H}$ may give the same polynomial, i.e., $f_{H_1}(\lambda) = f_{H_2}(\lambda)$. Let $\hat{\mathcal{H}}$ be a subset of $\mathcal{H}$ such that there is a bijection $g : \hat{\mathcal{H}} \to \mathsf{cp}$. Then

$$|\mathsf{cp}| = |\hat{\mathcal{H}}|.$$

For any two distinct subsets $H_1, H_2 \in \hat{\mathcal{H}}$, the degree-$wt$ polynomials $g(H_1) = f_{H_1}(\lambda)$ and $g(H_2) = f_{H_2}(\lambda)$ must be distinct. By Lemma 1, the number of indices $j \in [k]$ such that $f_{H_1}^{(\leq 1)}(\lambda_j) = f_{H_2}^{(\leq 1)}(\lambda_j)$ is upper bounded by $\lfloor wt/2 \rfloor$. Therefore, we must have that $|H_1 \cap H_2| \leq \lfloor wt/2 \rfloor$, and thus

$$|H_1 \setminus H_2| = |H_2 \setminus H_1| \geq k - b - \lfloor wt/2 \rfloor. \tag{14}$$

For every $H \in \hat{\mathcal{H}}$, we define a binary vector $c_H = (c_{H,1}, \ldots, c_{H,k})$ of length k such that

$$c_{H,j} = \begin{cases} 1, & j \in H, \\ 0, & j \notin H. \end{cases}$$

Then it is easy to see that c_H is of Hamming weight $k-b$. As per Eq. (14), the Hamming distance between c_{H_1} and c_{H_2} for any two distinct subsets $H_1, H_2 \in \hat{\mathcal{H}}$ is at least 2δ for

$$\delta = k - b - \lfloor wt/2 \rfloor.$$

Therefore, $C = \{c_H\}_{H \in \hat{\mathcal{H}}}$ is a binary $(k, 2\delta, k-b)$ constant weight code. Let $A(k, 2\delta, k-b)$ be the maximum size of a binary $(k, 2\delta, k-b)$ constant weight codes. As per Theorem 1, we have that

$$\begin{aligned} |\hat{\mathcal{H}}| = |\{c_H\}_{H \in \hat{\mathcal{H}}}| &\leq A(k, 2\delta, k-b) \\ &\leq \frac{\binom{k}{k-b-\delta+1}}{\binom{k-b}{k-b-\delta+1}} \\ &= \frac{\binom{k}{\lfloor wt/2 \rfloor+1}}{\binom{k-b}{\lfloor wt/2 \rfloor+1}} \\ &= O((\frac{k}{k-b})^{\lfloor wt/2 \rfloor+1}) \\ &= k^{O(1)}, \end{aligned}$$

i.e., the size L of $\mathsf{output\_list}$ is at most $k^{O(1)}$. □

t-Privacy. The t-privacy property of the proposed scheme is identical to that of the Woodruff-Yekhanin RPIR scheme, because both schemes share the same querying algorithm.

Communication complexity. The querying algorithm requires the client to send a vector $q_j \in \mathbb{F}_p^m$ to each of the ℓ servers. The answering algorithm requires each server $\mathcal{S}_j$ to send a vector $a_j = (u_j, \mathbf{v}_j) \in \mathbb{F}_p^{m+1}$ to the client. Therefore, the average number of bits exchanged for retrieving one bit from the database is $\ell(2m+1) = O(\ell m)$. As the integer m is chosen such that $\binom{m}{w} \geq n$, we have that $m = O(wn^{1/w})$ and thus

$$CC_{\Gamma_1}(n) = O(\ell w n^{1/w}).$$

Note that $n \gg w$ in a typical application scenario of PIR, therefore $CC_{\Gamma_1}(n)$ is a monotonically decreasing function of w. Ideally, we prefer to choose a large w in order to reduce the communication complexity. Theorem 2 stipulates that the list size $L = \left(\frac{k}{k-b}\right)^{\lfloor wt/2 \rfloor + 1}$. We present different strategies for choosing w, which correspond to different list sizes and communication complexities.

- When $\frac{k}{k-b} = O(1)$, that is, there exists a constant integer c such that $b < k - k/c$, we can choose $w = O\left(\frac{\log k}{t}\right)$. In this case, the communication complexity of our scheme Γ_1 is given by $CC_{\Gamma_1}(n) = \frac{\ell \log k}{t} n^{O(t/\log k)}$, and the list size is $L = k^{O(1)}$.
- When $\frac{k}{k-b} = O(k^c)$ for some constant $c \leq 1$, we can choose $w = O\left(\frac{1}{t}\right)$. In this case, the communication complexity of our scheme Γ_1 is given by $CC_{\Gamma_1}(n) = \frac{\ell}{t} n^{1/O(1/t)}$, and the list size is $L = k^{O(1)}$.

Notably, when $k \leq 2^5$, the differences between $O(k/t)$, $O(\log k/t)$, and $O(1/t)$ are not significant. In this case, we can directly set $w = \left\lfloor \frac{2k-2b-2}{t} \right\rfloor$. The communication complexity is

$$CC_{\Gamma_1}(n) = O\left(\ell(k-b) n^{1/\lfloor (2k-2b-2)/t \rfloor}\right),$$

and the list size is $L = \left(\frac{k}{k-b}\right)^{k-b}$. For the ease of sublinear communication complexity, we require that $b \leq k-2$.

4.3 Improving the Client-Side Computation Complexity

In Step 3 of reconstructing algorithm $\Gamma_1.\mathcal{R}$, the client interpolates a polynomial with the set H for any $H \subset [k]$ of cardinality $k-b$, which incurs the $\binom{k}{k-b}$ interpolations. In this section, we propose a method to reduce the number of interpolations to $\binom{k}{wt/2}$. Since $wt < 2(k-b)-1$ and $wt = O(1)$ with respect to the parameter k, the client-side computational complexity is significantly reduced. Specifically, the complexity decreases from exponential in k to a polynomial function of k.

We replace the Step 3 of $\Gamma_1.\mathcal{R}$ with the following step:

3. Initialize a set $\mathsf{cp} = \emptyset$ to store the candidate polynomials. For any set $H \subseteq [k]$ of cardinality $\lfloor wt/2 \rfloor + 1$,

(3.1) Interpolate a polynomial $f_H(\lambda)$ such that for each $j \in H$, $f_H(\lambda_j) = \alpha_j, f'_H(\lambda_j) = \beta_j$.
(3.2) If $f_H(\lambda)$ is of degree at most wt and there exist at least $k - b$ distinct values of $j \in [k]$ satisfying $f(\lambda_j) = \alpha_j, f'(\lambda_j) = \beta_j$ add $f_H(\lambda)$ to the set cp.

To see why this substitution works, consider any polynomial $f_H(\lambda) \in \mathsf{cp}$ constructed in the original Step 3 of $\Gamma_2.\mathcal{R}$ for a set $H \subseteq [k]$ of cardinality $k-b$. Since $f_H(\lambda)$ is included in cp, its degree must not exceed wt. Now, consider any subset $H' \subseteq H$ with cardinality $\lfloor wt/2 \rfloor + 1$. The polynomial $f_{H'}(\lambda)$, which satisfies $f^{(\leq 1)}_{H'}(\lambda_j) = (\alpha_j, \beta_j)$ and has degree at most $wt + 1$, is uniquely determined as $f_{H'}(\lambda) = f_H(\lambda)$. The proof can be established by contradiction and is omitted here for brevity. Consequently, the polynomial $f_H(\lambda)$ will also be included in the set cp under the new Step 3.

With the substitution of Step 3, we can reduce the client-side computation complexity of $\Gamma_1.\mathcal{R}$ to be polynomial in k.

5 Perfect ldBRPIR based on weighted-degree polynomial

In this section, we construct a perfectly correct (L, b, t, k, ℓ)-ldBRPIR scheme Γ_2 with list size $L = O(k)$ and Byzantine robustness $b \leq k - \sqrt{kt}$. Compared with Γ_1, the scheme Γ_2 demonstrates reduced communication complexity when dealing with a larger number of servers.

5.1 The Construction

Same as Γ_1, the scheme Γ_2 retains the querying and answering algorithms of the Woodruff-Yekhanin RPIR scheme Γ, $(\Gamma_2.\mathcal{Q}, \Gamma_2.\mathcal{A}) = (\Gamma.\mathcal{Q}, \Gamma.\mathcal{A})$. The key differences between Γ_2 and Γ are as follows: **(I)** The degree parameter w is required to satisfy Eq. (3) in Γ, whereas in Γ_2 it is required to satisfy that $w \leq (k-b)^2/t$. **(II)** The reconstructing algorithm $\Gamma.\mathcal{R}$ in Γ is replaced by the reconstructing algorithm $\Gamma_2.\mathcal{R}$ in Γ_2, as shown in **Fig** 3.

In scheme Γ_2, we achieve the expected list size and Byzantine robustness by extending Sudan's list-decoding algorithm for Reed-Solomon codes to the order-1 univariate multiplicity codes. Each responding server $\mathcal{S}_j (j \in [k])$ in the proposed scheme Γ_2 returns both an evaluation of the polynomial F that encodes the database $\mathbf{x}$ and m partial derivatives of F, all computed at the same query point $q_j \in \mathbb{F}_p^m$. The response of each server $\mathcal{S}_j$ gives an order-1 evaluation $f(\lambda) = F(G(\lambda))$ at a field element λ_j, i.e., $f^{(\leq 1)}(\lambda_j)$. When at most b out of the k responding servers are malicious and respond incorrectly, the problem of constructing a list that contains $x_i = f(0)$ can be reduced to the problem of building a list that contains $f(\lambda)$, which is exactly the problem of list decoding an order-1 univariate multiplicity code for $f(\lambda)$ from the (corrupted) codeword $\{(\alpha_j, \beta_j)\}_{j=1}^k$, where $(\alpha_j, \beta_j) = f^{(\leq 1)}(\lambda_j)$ for any honest server $\mathcal{S}_j$ but may not be true for a malicious server.

$\mathsf{output\_list} \leftarrow \Gamma_2.\mathcal{R}(i, \{a_j\}_{j\in[\ell]}, \mathsf{aux})$:

1. Parse $\mathsf{aux} = (\{\lambda_j\}_{j\in[\ell]}, \{\mathbf{r}_s\}_{s\in[t]})$. Construct the polynomial $G(\lambda) = E(i) + \sum_{s=1}^{t} \lambda^s \mathbf{r}_s$.
2. Wlog, suppose the first k servers respond and denote $a_j = (u_j, \mathbf{v}_j)$ for all $j \in [k]$. For each $j \in [k]$, let $\alpha_j = u_j, \beta_j = \langle \mathbf{v}_j, G'(\lambda_j)\rangle$.
3. Let $D = 2(k-b) - 1$, $\rho = \lfloor \frac{D}{wt} \rfloor$. Define $\rho + 1$ fundamental polynomials $Q_0(\lambda), \ldots, Q_\rho(\lambda)$ and set

$$Q^{\text{base}}(\lambda, \alpha) = \sum_{s=0}^{\rho} Q_s(\lambda)\alpha^s,$$

$$Q^{\text{ext}}(\lambda, \alpha, \beta) = \sum_{s=1}^{\rho} s \cdot Q_s(\lambda)\alpha^{s-1}\beta + \sum_{s=0}^{\rho} Q'_s(\lambda)\alpha^s$$

such that:
(1) Q^{base} has a $(1, wt)$-weighted degree of at most D.
(2) For every $j \in [k]$,

$$Q^{\text{base}}(\lambda_j, \alpha_j) = 0, \tag{15}$$

$$Q^{\text{ext}}(\lambda_j, \alpha_j, \beta_j) = 0. \tag{16}$$

4. Factor $Q(\lambda, \alpha)$ into irreducible factors. Define the set of candidate polynomials cp as the set of degree-wt polynomials $\tilde{f}(\lambda)$ such that $\alpha - \tilde{f}(\lambda)$ is a factor of $Q^{\text{base}}(\lambda, \alpha)$ and there are at least $k - b$ distinct $j \in [k]$ such that $\tilde{f}(\lambda_j) = \alpha_j$ and $\tilde{f}'(\lambda_j) = \beta_j$.
5. Output $\mathsf{output\_list} = \{\tilde{f}(0) : \tilde{f}(\lambda) \in \mathsf{cp}\}$.

Fig. 3. Reconstructing algorithm $\Gamma_2.\mathcal{R}$ of the (L, t, k, ℓ, b)-ldBRPIR scheme Γ_2.

While Sudan's list decoding algorithm [34] enables one to correct a relatively large fraction of errors but is only applicable to order-0 univariate multiplicity codes (i.e., Reed-Solomon codes) and the recent list decoding algorithms [21] are applicable to general order-s univariate multiplicity codes but only allow recovery from a relatively small fraction of errors, the main innovative idea underlying Γ_2 is achieving the (possibly) best from both worlds by injecting Sudan's idea of using the non-linear terms α^c in the weighted-degree bivariate polynomial Q of Eq. (1) into the construction of the multivariate polynomial Q of Eq. (2), which is linear in $\alpha_0, \ldots, \alpha_s$. To realize this idea, we simply replace the linear term α_c in Eq. (2) with its properly selected higher degree powers. The main observation about this simple modification to Eq. (2) is that the new polynomial may accommodate a much larger number of monomials and thus significantly improve the fraction of correctable errors.

We denote by $Q^{\text{base}}(\lambda, \alpha)$ the new polynomial resulted from the aforementioned idea and denote by $Q^{\text{ext}}(\lambda, \alpha, \beta)$ the polynomial obtained by applying the special operator $\mathcal{D}$ from Section 2.3 to $Q^{\text{base}}(\lambda, \alpha)$, where $\beta = \mathcal{D}(\alpha)$. More

precisely, for $\rho = \lfloor \frac{D}{wt} \rfloor$, we choose a bivariate base polynomial

$$Q^{\text{base}}(\lambda, \alpha) = \sum_{s=0}^{\rho} Q_s(\lambda)\alpha^s. \tag{17}$$

and impose k constraints of the form

$$Q^{\text{base}}(\lambda_j, \alpha_j) = 0$$

for all $j \in [k]$. We calculate the trivariate extended polynomial

$$Q^{\text{ext}}(\lambda, \alpha, \beta) = \mathcal{D}(Q^{\text{base}}) = \sum_{s=1}^{\rho} s \cdot Q_s(\lambda)\alpha^{s-1}\beta + \sum_{s=0}^{\rho} Q'_s(\lambda)\alpha^s$$

and impose k additional constraints of the form

$$Q^{\text{ext}}(\lambda_j, \alpha_j, \beta_j) = 0$$

for all $j \in [k]$.

To better understand the roles of the polynomials $\{Q_s(\lambda)\}_{s=0}^{\rho}$, consider the polynomial $p(\lambda) = Q^{\text{base}}(\lambda, \tilde{f}(\lambda))$, where $\tilde{f}(\lambda)$ is a polynomial of degree $\leq wt$ such that $\tilde{f}^{(\leq 1)}(\lambda_j) = (\alpha_j, \beta_j)$ for at least $k - b$ distinct indices $j \in [k]$. If $j \in [k]$ is any index such that $\tilde{f}^{(\leq 1)}(\lambda_j) = (\alpha_j, \beta_j)$, then we would have

$$\begin{aligned} p(\lambda_j) &= Q^{\text{base}}(\lambda_j, \tilde{f}(\lambda_j)) = 0; \\ p'(\lambda_j) &= Q^{\text{ext}}(\lambda_j, \tilde{f}(\lambda_j), \tilde{f}'(\lambda_j)) = 0. \end{aligned}$$

Hence, the polynomial $p(\lambda)$ should be of degree $\geq 2(k - b)$ if it is nonzero. By choosing a weighted degree parameter $D = 2(k - b) - 1$ and properly choosing degrees for the polynomials $\{Q_s(\lambda)\}_{s=0}^{\rho}$, we force $p(\lambda)$ to be a polynomial of degree $\leq D$ and thus identically zero, which in turn enable use to extract $\tilde{f}(\lambda)$ by factoring the interpolated polynomial $Q^{\text{base}}(\lambda, \alpha)$.

5.2 Analysis

In this section, we show that the proposed scheme Γ_2 is indeed an (L, b, t, k, ℓ)-ldBRPIR scheme for $L = O(k)$ and $b \leq k - \sqrt{kt}$.

Correctness. To show that the proposed scheme is correct, we firstly identify under what conditions there do exist $\rho + 1$ polynomials $\{Q_s(\lambda)\}_{s=0}^{\rho}$ that satisfy the the constraints of Eq. (15) and (16). Such conditions may guide our selections of parameters.

Lemma 2. *If $\rho = \lfloor D/wt \rfloor$ and $w \leq \frac{(D+1)^2}{4kt}$ $\left(= \frac{(k-b)^2}{kt}\right)$, there exist $\rho + 1$ polynomials $\{Q_s(\lambda)\}_{s=0}^{\rho}$ that satisfy the constraints imposed by Eq. (15) and (16), where $\deg(Q_s) \leq D - swt$ for all $s = 0, 1, \ldots, \rho$.*

Proof. Consider the bivariate polynomial $Q^{\text{base}}(\lambda, \alpha)$ of Eq. (17). If we choose $D = 2(k-b)-1$ and choose every $Q_s(\lambda)$ there to be a polynomial of degree $\leq D - swt$, then the $(1, wt)$-weighted degree of $Q^{\text{base}}(\lambda, \alpha)$ is $\leq D$. Let num be the number of monomials in $Q^{\text{base}}(\lambda, \alpha)$. Given that $\rho = \lfloor D/wt \rfloor$, we have

$$\mathsf{num} = \sum_{s=0}^{\rho} (D - swt + 1) \geq \frac{(D+1)^2}{2wt} + 1.$$

On the other hand, the number of constraints imposed by Eq. (15) and (16) is $2k$. When $w \leq \frac{(D+1)^2}{4kt}$, we have that

$$2k \leq \frac{(D+1)^2}{2wt} < \mathsf{num}.$$

As the number of coefficients in $Q^{\text{base}}(\lambda, \alpha)$ exceeds the number of constraints given by Eq. (15) and (16), there must exist a nonzero bivariate polynomial $Q^{\text{base}}(\lambda, \alpha)$ that satisfies all of the $2k$ constraints. The existence of $Q^{\text{base}}(\lambda, \alpha)$ implies that of $Q_0(\lambda), \ldots, Q_\rho(\lambda)$. □

Lemma 2 shows the existence of the polynomials $Q_0, ..., Q_\rho$ when we properly choose the parameters in Γ_2. Since w is the degree of the polynomial $F(\mathbf{z})$ encoding the database $\mathbf{x}$, we must have that $w \geq 1$, which together with the condition $w \leq (k-b)^2/(kt)$ implies that $b \leq k - \sqrt{kt}$, i.e., the scheme Γ_2 can tolerate as many as $k - \sqrt{kt}$ malicious servers that respond incorrectly.

Theorem 3. *The scheme Γ_2 is correct with list size $L = O(k)$ when $1 \leq w \leq \frac{(k-b)^2}{kt}$.*

Proof. Let $\mathsf{output\_list}$ be the list output by $\Gamma_2.\mathcal{R}$. It suffices to show that **(I)** $x_i \in \mathsf{output\_list}$; and **(II)** $|\mathsf{output\_list}| \leq O(k)$, *under the proposed choices of the parameters w, b, t and k.*

(I) Referring to the description of $\Gamma_2.\mathcal{R}$ in Figure 3, wlog the first k servers respond. Consider the degree-wt polynomial

$$f(\lambda) = F(G(\lambda)) = F(E(i) + \sum_{s=1}^{t} \lambda^s \mathbf{r}_s).$$

For every $j \in [k]$, it is easy to see that

$$\begin{aligned} f(\lambda_j) &= F(G(\lambda_j)) = F(q_j), \\ f'(\lambda_j) &= \sum_{c=1}^{m} \left.\frac{\partial F(\mathbf{z})}{\partial z_c}\right|_{G(\lambda_j)} \cdot G'(\lambda_j)_c \\ &= \langle \mathbf{v}_j, G'(\lambda_j) \rangle, \end{aligned}$$

where $G'(\lambda_j)_c$ denotes the c-th entry of the vector $G'(\lambda_j)$. Consider the polynomial $p(\lambda) = Q^{\text{base}}(\lambda, f(\lambda))$. Since $f(\lambda)$ is a polynomial of degree wt and the

bivariate polynomial $Q^{\text{base}}(\lambda, \alpha)$ has a $(1, wt)$-weighted degree $\leq D$, the degree of $p(\lambda)$ is $\leq D$. For any $j \in [k]$ and any triplet $(\lambda_j, \alpha_j, \beta_j)$ such that $f^{(\leq 1)}(\lambda_j) = (\alpha_j, \beta_j)$, we have

$$\begin{aligned} p(\lambda_j) &= Q^{\text{base}}(\lambda_j, f(\lambda_j)) \\ &= Q^{\text{base}}(\lambda_j, \alpha_j) \\ &= 0, \\ p'(\lambda_j) &= \sum_{s=1}^{\rho} s \cdot Q_s(\lambda_j) f(\lambda_j)^{s-1} f'(\lambda_j) + \sum_{s=0}^{\rho} Q'_s(\lambda_j) f(\lambda_j)^s \\ &= Q^{\text{ext}}(\lambda_j, f(\lambda_j), f'(\lambda_j)) \\ &= Q^{\text{ext}}(\lambda_j, \alpha_j, \beta_j) \\ &= 0. \end{aligned}$$

Thus, the polynomial $p(\lambda)$ should be of degree $\geq 2(k-b)$, if it is nonzero. On the other hand, our choices of all parameters imply that $\deg(p(\lambda)) \leq D = 2(k-b)-1$. Hence, $p(\lambda)$ must be the zero polynomial, i.e., $Q^{\text{base}}(\lambda, f(\lambda)) = 0$. It follows that $(\alpha - f(\lambda)) \mid Q^{\text{base}}(\lambda, \alpha)$ Thus, $f(\lambda)$ will be included in the set cp of candidate polynomials can cause $f(0) = x_i$ to be included into the set $\mathsf{output\_list}$.

(II) Referring to the description of $\Gamma_2.\mathcal{R}$ in Figure 3, for every $\tilde{f}(\lambda) \in \mathsf{cp}$, the polynomial $\alpha - \tilde{f}(\lambda)$ must be a linear factor of $Q^{\text{base}}(\lambda, \alpha)$. As $Q^{\text{base}}(\lambda, \alpha)$ is a polynomial of degree $\leq \rho$ in α, the number of polynomials in cp must be at most ρ, i.e., $|\mathsf{cp}| \leq \rho$. Consequently, we have that

$$L = |\mathsf{output\_list}| \leq |\mathsf{cp}| \leq \rho = \lfloor \frac{2(k-b)-1}{wt} \rfloor = O(k).$$

In particular, we have that $L \leq 2k$ for any choices of (k, b, t). Furthermore, when the majority of the server responses are incorrect, that is, $b > k/2$, we have $L \leq k$. □

t-Privacy. The t-privacy property of the proposed scheme is identical to that of the Woodruff-Yekhanin RPIR scheme and our scheme Γ_1, because those schemes share the same querying algorithm.

Communication complexity. Same as scheme Γ_1, we have $m = O(wn^{1/w})$ in scheme Γ_2 and the communication complexity is

$$CC_{\Gamma_2}(n) = O(\ell w n^{1/w}).$$

Note that $n \gg w$ in a typical application scenario of PIR, therefore $CC_{\Gamma_2}(n)$ is a monotonically decreasing function of w. Ideally, we prefer to choose a large w in order to reduce the communication complexity. Theorem 3 establishes a scaled upper bound for w, specifically $w \leq \frac{(k-b)^2}{kt}$, implying that the communication complexity $CC_{\Gamma_2}(n)$ of our scheme Γ_2 is

$$CC_{\Gamma_2}(n) = O\left(\frac{\ell(k-b)^2}{kt} n^{1/\lfloor \frac{(k-b)^2}{kt} \rfloor} \right).$$

Our scheme Γ_2 shares the same Byzantine tolerance bound $b \leq k - \sqrt{kt}$ as the scheme in [16]. For different numbers of malicious servers b, the relationship between the communication complexity $CC_{\Gamma_2}(n)$ of scheme Γ_2 and the database size n is compared with the schemes in [16] and [12] as follows:

- When $k-\sqrt{2kt} < b \leq k-\sqrt{kt}$, the communication complexity of our scheme Γ_2 is $O(n)$, which is less efficient than the communication complexity of the schemes in both [16] and [12], achieving $O(n^{1/2})$.
- When $k-\sqrt{3kt} < b \leq k-\sqrt{2kt}$, the communication complexity of our scheme Γ_2 is $O(n^{1/2})$, matching the efficiency of the schemes in [16] and [12].
- When $b \leq k - \sqrt{3kt}$, the communication complexity of our scheme Γ_2 can achieve $o(n^{1/2})$, offering a significant improvement over the schemes in [16] and [12].

6 Implementation

We implemented the scheme Γ_1 and Γ_2 presented in this paper using C++ and the FLINT library [36] and compared the efficiency of the scheme described in [16] and [12]. The database used in the experiments was generated with a custom Python-based generator. Our implementation of the schemes and the database generator is publicly available on GitHub. We have uploaded only a smaller example database and Larger databases can be reproduced using the generator provided in our repository.

We measure the performance of our scheme on a computer with a 16-core intel Xeon Gold 6250 CPU 3.90GHz and 256 GB RAM, running Ubuntu 20.04. All of our experiments are single-threaded.

In the foregoing sections, we scrutinized the Byzantine robustness of Γ_1 and Γ_2. In our experiments, we focus on evaluating **(1)** the communication volume exchanged between the client and the servers; and **(2)** the size of the output lists generated by the schemes.

(1) To evaluate the communication volume between the client and servers, we analyze the total size of the queries $\{q_j\}_{j\in[\ell]}$ generated by the querying algorithm and the responses $\{a_j\}_{j\in[\ell]}$ produced by the answering algorithm.

(2) To determine the output list size, we count the number of elements in the list $\mathsf{output\_list}$ produced by the reconstructing algorithm. For different databases $\mathbf{x}$, indices i of interest to the client, chosen queries $\{q_j\}_{j\in[\ell]}$, and adversarial responses $\{\hat{a}_j\}_{j\in[b]}$ from Byzantine servers, the size of the output list from the reconstructing algorithm varies. Moreover, it is challenging to identify a specific combination of $\mathbf{x}$, i, $\{q_j\}_{j\in[\ell]}$, and $\{\hat{a}_j\}_{j\in[b]}$ that maximizes the output list size. To address this, we fix modest values for the database size n, the field size p, and the parameters (k, b, t), then allow the adversary to inject random erroneous responses. We execute the protocol repeatedly and record the worst-case list size yielded by the reconstructing algorithm.

Scheme	Vary k			Vary b		
	k	Asymptotic	Experimental	b	Asymptotic	Experimental
Γ_1	16	$O(n^{1/4})$	20.0 ± 0.1 MiB	10	$O(n^{1/16})$	735.5 ± 20 KiB
	18	$O(n^{1/8})$	2.2 ± 0.1 MiB	11	$O(n^{1/14})$	821.3 ± 20 KiB
	20	$O(n^{1/12})$	1.0 ± 0.1 MiB	12	$O(n^{1/12})$	1.0 ± 0.1 MiB
	22	$O(n^{1/16})$	732.2 ± 20 KiB	13	$O(n^{1/10})$	1.4 ± 0.1 MiB
	24	$O(n^{1/20})$	612.5 ± 20 KiB	14	$O(n^{1/8})$	2.2 ± 0.1 MiB
Γ_2	16	$O(n)$	Too large	10	$O(n^{1/6})$	1.1 ± 0.1 MiB
	18	$O(n^{1/2})$	325.8 ± 5 MiB	11	$O(n^{1/5})$	1.5 ± 0.1 MiB
	20	$O(n^{1/4})$	3.6 ± 0.1 MiB	12	$O(n^{1/4})$	3.6 ± 0.1 MiB
	22	$O(n^{1/5})$	1.5 ± 0.1 MiB	13	$O(n^{1/3})$	15.3 ± 0.2 MiB
	24	$O(n^{1/8})$	421.5 ± 20 KiB	14	$O(n^{1/2})$	317.2 ± 5 MiB
[16]	16	/	/	10	$O(n^{1/2})$	12.8 ± 0.1 MiB
	18	$O(n^{1/2})$	12.8 ± 0.1 MiB	11	$O(n^{1/2})$	12.8 ± 0.1 MiB
	20	$O(n^{1/2})$	12.8 ± 0.1 MiB	12	$O(n^{1/2})$	12.8 ± 0.1 MiB
	22	$O(n^{1/2})$	12.8 ± 0.1 MiB	13	$O(n^{1/2})$	12.8 ± 0.1 MiB
	24	$O(n^{1/2})$	12.8 ± 0.1 MiB	14	$O(n^{1/2})$	12.8 ± 0.1 MiB
[12]	16	$O(n^{1/2})$	12.8 ± 0.1 MiB	10	$O(n^{1/2})$	12.8 ± 0.1 MiB
	18	$O(n^{1/2})$	12.8 ± 0.1 MiB	11	$O(n^{1/2})$	12.8 ± 0.1 MiB
	20	$O(n^{1/2})$	12.8 ± 0.1 MiB	12	$O(n^{1/2})$	12.8 ± 0.1 MiB
	22	$O(n^{1/2})$	12.8 ± 0.1 MiB	13	$O(n^{1/2})$	12.8 ± 0.1 MiB
	24	$O(n^{1/2})$	12.8 ± 0.1 MiB	14	$O(n^{1/2})$	12.8 ± 0.1 MiB

Scheme	Vary t			Vary $\mathbb{F}_p$		
	t	Asymptotic	Experimental	$\log \lvert\mathbb{F}_p\rvert$	Asymptotic	Experimental
Γ_1	1	$O(n^{1/12})$	1.0 ± 0.1 MiB	16	$O(n^{1/12})$	152 ± 10 KiB
	2	$O(n^{1/6})$	3.2 ± 0.1 MiB	32	$O(n^{1/12})$	267 ± 10 KiB
	3	$O(n^{1/4})$	11.5 ± 0.1 MiB	64	$O(n^{1/12})$	508 ± 10 KiB
	4	$O(n^{1/3})$	41.0 ± 0.3 MiB	128	$O(n^{1/12})$	1.0 ± 0.1 MiB
Γ_2	1	$O(n^{1/4})$	3.6 ± 0.1 MiB	16	$O(n^{1/4})$	465 ± 10 KiB
	2	$O(n^{1/2})$	315.4 ± 5 MiB	32	$O(n^{1/4})$	932 ± 20 KiB
	3	$O(n)$	Too large	64	$O(n^{1/4})$	1.8 ± 0.1 MiB
	4	$O(n)$	Too large	128	$O(n^{1/4})$	3.6 ± 0.1 MiB
[16]	1	$O(n^{1/2})$	12.8 ± 0.1 MiB	16	$O(n^{1/2})$	1.6 ± 0.1 MiB
	2	$O(n^{1/2})$	12.8 ± 0.1 MiB	32	$O(n^{1/2})$	3.2 ± 0.1 MiB
	3	$O(n^{1/2})$	12.8 ± 0.1 MiB	64	$O(n^{1/2})$	6.4 ± 0.1 MiB
	4	$O(n^{1/2})$	12.8 ± 0.1 MiB	128	$O(n^{1/2})$	12.8 ± 0.1 MiB
[12]	1	$O(n^{1/2})$	12.8 ± 0.1 MiB	16	$O(n^{1/2})$	1.6 ± 0.1 MiB
	2	$O(n^{1/2})$	12.8 ± 0.1 MiB	32	$O(n^{1/2})$	3.2 ± 0.1 MiB
	3	$O(n^{1/2})$	12.8 ± 0.1 MiB	64	$O(n^{1/2})$	6.4 ± 0.1 MiB
	4	$O(n^{1/2})$	12.8 ± 0.1 MiB	128	$O(n^{1/2})$	12.8 ± 0.1 MiB

Table 2. Asymptotic and Experimental Per-Server Communication Complexities under Variations of k, b, t, and $\log |\mathbb{F}_p|$, Averaged over 100 Trials. The Baseline Configuration is $(k, b, t, \log |\mathbb{F}_p|) = (20, 12, 1, 128)$.

Scheme	$\log \lvert\mathbb{F}_p\rvert$	Worst List Size	$\log \lvert\mathbb{F}_p\rvert$	Worst List Size
Γ_1	7	2	10	2
Γ_2	7	3	10	4
[16]	7	8	10	15
[12]	7	6	10	13

Table 3. Worst-case output list sizes (over 10^6 runs) for four ldBRPIR schemes with parameters $(k, b, t) = (6, 3, 1)$ and database size $n = 2^{16}$, instantiated over two prime fields of size $p = 2^7 + 3$ and $2^{10} + 7$ (i.e. $\log |\mathbb{F}_p| = 7$ and 10).

The experiments are divided into two phases. In Phase I, we employ a database of size $n = 2^{26}$ over $\mathbb{F}_p$, with initial parameters $(k, b, t) = (20, 12, 1)$ and $p = 2^{128} + 51$. Thereafter, we individually vary k, b, t, and $\log p$, derive the asymptotic communication complexity per server, and measure the experimental communication volume. Throughout, for Γ_1 we set $w_1 = 2(k - b - 2)/t$. To minimize errors caused by randomness, each experiment is repeated 100 times, and the average results are recorded as in **Table.** 2. As evidenced by the data, although schemes Γ_1 and Γ_2 fall short of the protocols in [16] and [12] for larger values of t, they incur markedly lower communication overhead when t is small. Moreover, this benefit grows more pronounced as the total number of servers increases and the number of Byzantine servers decreases. In Phase II, we consider the database of size $n = 2^{16}$ over $\mathbb{F}_p$, with parameters $(k, b, t) = (6, 3, 1)$. We instantiate two variants over distinct prime fields, namely $p = 2^7 + 3$ and $p = 2^{10} + 7$. For each configuration, the scheme is executed 10^6 times, and the worst-case cardinality of the output list is reported in **Table.** 3. It is evident that the maximum list sizes of schemes Γ_1 and Γ_2 remain small and exhibit negligible variation as the field $\mathbb{F}_p$ grows. In contrast, the protocols of [16] and [12] display a marked increase in list size. Consequently, one can surmise that for sufficiently large $\mathbb{F}_p$ there exist parameter choices (k, b, t) and adversarial response strategies under which those earlier schemes yield substantially larger lists, whereas our constructions continue to produce output lists whose size remains bounded independently of p.

7 Conclusion

We have introduced two perfect list-decodable BRPIR schemes, Γ_1 and Γ_2. The Byzantine tolerance bound of scheme Γ_1 significantly surpasses those established in [16] and [12]. Compared to the schemes proposed in [16] and [12], our schemes demonstrate a notable improvement in communication complexity. This advantage is particularly pronounced in scenarios with a small and fixed privacy threshold, such as $t = O(1)$. We achieve a small maximum output list size that is independent of the size of the finite field containing the data and depends solely on the number of servers. Additionally, our scheme Γ_1 offers higher Byzantine robustness than [16] and [12].

References

1. Agrell, E., Vardy, A., Zeger, K.: Upper bounds for constant-weight codes. IEEE Transactions on Information Theory **46**(7), 2373–2395 (2000)
2. Angel, S., Chen, H., Laine, K., Setty, S.: Pir with compressed queries and amortized query processing. In: 2018 IEEE symposium on security and privacy (SP). pp. 962–979. IEEE (2018)
3. Angel, S., Setty, S.: Unobservable communication over fully untrusted infrastructure. In: 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16). pp. 551–569 (2016)
4. Banawan, K., Ulukus, S.: The capacity of private information retrieval from byzantine and colluding databases. IEEE Transactions on Information Theory **65**(2), 1206–1219 (2018)
5. Beimel, A., Ishai, Y.: Information-theoretic private information retrieval: A unified construction. In: Automata, Languages and Programming: 28th International Colloquium, ICALP 2001 Crete, Greece, July 8–12, 2001 Proceedings 28. pp. 912–926. Springer (2001)
6. Beimel, A., Stahl, Y.: Robust information-theoretic private information retrieval. In: International Conference on Security in Communication Networks. pp. 326–341. Springer (2002)
7. Bose, Rao: Theory of unidirectional error correcting/detecting codes. IEEE Transactions on Computers **100**(6), 521–530 (1982)
8. Cao, Q., Tran, H.Y., Dau, S.H., Yi, X., Viterbo, E., Feng, C., Huang, Y.C., Zhu, J., Kruglik, S., Kiah, H.M.: Committed private information retrieval. In: European Symposium on Research in Computer Security. pp. 393–413. Springer (2023)
9. Chee, Y.M., Feng, T., Ling, S., Wang, H., Zhang, L.F.: Query-efficient locally decodable codes of subexponential length. computational complexity **22**(1), 159–189 (2013)
10. Chor, B., Kushilevitz, E., Goldreich, O., Sudan, M.: Private information retrieval. Journal of the ACM (JACM) **45**(6), 965–981 (1998)
11. Colombo, S., Nikitin, K., Corrigan-Gibbs, H., Wu, D.J., Ford, B.: Authenticated private information retrieval. In: 32nd USENIX security symposium (USENIX Security 23). pp. 3835–3851 (2023)
12. Devet, C., Goldberg, I., Heninger, N.: Optimally robust private information retrieval. In: 21st USENIX Security Symposium (USENIX Security 12). pp. 269–283 (2012)
13. Dvir, Z., Gopi, S.: 2-server pir with subpolynomial communication. Journal of the ACM (JACM) **63**(4), 1–15 (2016)
14. Efremenko, K.: 3-query locally decodable codes of subexponential length. In: Proceedings of the forty-first annual ACM symposium on Theory of computing. pp. 39–44 (2009)
15. Eriguchi, R., Kurosawa, K., Nuida, K.: Multi-server pir with full error detection and limited error correction. Cryptology ePrint Archive (2022)
16. Goldberg, I.: Improving the robustness of private information retrieval. In: 2007 IEEE Symposium on Security and Privacy (SP'07). pp. 131–148. IEEE (2007)
17. Goyal, R., Harsha, P., Kumar, M., Shankar, A.: Fast list decoding of univariate multiplicity and folded reed-solomon codes. In: 2024 IEEE 65th Annual Symposium on Foundations of Computer Science (FOCS). pp. 328–343. IEEE (2024)
18. Gupta, T., Crooks, N., Mulhern, W., Setty, S., Alvisi, L., Walfish, M.: Scalable and private media consumption with popcorn. In: 13th USENIX symposium on networked systems design and implementation (NSDI 16). pp. 91–107 (2016)

19. Guruswami, V., Rudra, A.: Explicit codes achieving list decoding capacity: Error-correction with optimal redundancy. IEEE Transactions on information theory **54**(1), 135–150 (2008)
20. Guruswami, V., Sudan, M.: Improved decoding of reed-solomon and algebraic-geometric codes. In: Proceedings 39th Annual Symposium on Foundations of Computer Science (Cat. No. 98CB36280). pp. 28–37. IEEE (1998)
21. Guruswami, V., Wang, C.: Optimal rate list decoding via derivative codes. In: International Workshop on Approximation Algorithms for Combinatorial Optimization. pp. 593–604. Springer (2011)
22. Guruswami, V., Wang, C.: Linear-algebraic list decoding for variants of reed–solomon codes. IEEE Transactions on Information Theory **59**(6), 3257–3268 (2013)
23. Ke, P., Zhang, L.F.: Two-server private information retrieval with result verification. In: 2022 IEEE International Symposium on Information Theory (ISIT). pp. 408–413. IEEE (2022)
24. Ke, P., Zhang, L.F.: Private information retrieval with result verification for more servers. In: International Conference on Applied Cryptography and Network Security. pp. 197–216. Springer (2023)
25. Khoshgozaran, A., Shahabi, C.: Private information retrieval techniques for enabling location privacy in location-based services. Privacy in Location-Based Applications: Research Issues and Emerging Trends pp. 59–83 (2009)
26. Kopparty, S.: List-decoding multiplicity codes. Theory of Computing **11**(1), 149–182 (2015)
27. Kopparty, S., Ron-Zewi, N., Saraf, S., Wootters, M.: Improved list decoding of folded reed-solomon and multiplicity codes. SIAM Journal on Computing **52**(3), 794–840 (2023)
28. Kruglik, S., Dau, S.H., Kiah, H.M., Wang, H., Zhang, L.F.: Querying twice to achieve information-theoretic verifiability in private information retrieval. Authorea Preprints (2023)
29. Kurosawa, K.: How to correct errors in multi-server pir. In: International Conference on the Theory and Application of Cryptology and Information Security. pp. 564–574. Springer (2019)
30. Kushilevitz, E., Ostrovsky, R.: Replication is not needed: Single database, computationally-private information retrieval. In: Proceedings 38th annual symposium on foundations of computer science. pp. 364–373. IEEE (1997)
31. Melchor, C.A., Barrier, J., Fousse, L., Killijian, M.O.: Xpir: Private information retrieval for everyone. Proceedings on Privacy Enhancing Technologies pp. 155–174 (2016)
32. Patel, S., Persiano, G., Yeo, K.: Private stateful information retrieval. In: Proceedings of the 2018 ACM SIGSAC conference on computer and communications security. pp. 1002–1019 (2018)
33. Shamir, A.: How to share a secret. Communications of the ACM **22**(11), 612–613 (1979)
34. Sudan, M.: Decoding of reed solomon codes beyond the error-correction bound. Journal of complexity **13**(1), 180–193 (1997)
35. Tajeddine, R., Gnilke, O.W., Karpuk, D., Freij-Hollanti, R., Hollanti, C.: Private information retrieval from coded storage systems with colluding, byzantine, and unresponsive servers. IEEE Transactions on information theory **65**(6), 3898–3906 (2019)
36. team, T.F.: FLINT: Fast Library for Number Theory (2023), version 3.0.0, `https://flintlib.org`

37. Welch, L.R., Berlekamp, E.R.: Error correction for algebraic block codes (1986)
38. Woodruff, D., Yekhanin, S.: A geometric approach to information-theoretic private information retrieval. In: 20th Annual IEEE Conference on Computational Complexity (CCC'05). pp. 275–284. IEEE (2005)
39. Yannuzzi, M., Milito, R., Serral-Gracià, R., Montero, D., Nemirovsky, M.: Key ingredients in an iot recipe: Fog computing, cloud computing, and more fog computing. In: 2014 IEEE 19th international workshop on computer aided modeling and design of communication links and networks (CAMAD). pp. 325–329. IEEE (2014)
40. Yekhanin, S.: Towards 3-query locally decodable codes of subexponential length. Journal of the ACM (JACM) **55**(1), 1–16 (2008)
41. Zhang, L.F., Safavi-Naini, R.: Verifiable multi-server private information retrieval. In: International Conference on Applied Cryptography and Network Security. pp. 62–79. Springer (2014)
42. Zhang, L.F., Wang, H., Wang, L.P.: Byzantine-robust private information retrieval with low communication and efficient decoding. In: Proceedings of the 2022 ACM on Asia Conference on Computer and Communications Security. pp. 1079–1085 (2022)
43. Zhao, L., Wang, X., Huang, X.: Verifiable single-server private information retrieval from lwe with binary errors. Information Sciences **546**, 897–923 (2021)
44. Zhu, L., Lin, C., Lin, F., Zhang, L.F.: Post-quantum cheating detectable private information retrieval. In: IFIP International Conference on ICT Systems Security and Privacy Protection. pp. 431–448. Springer (2022)