# List Decoding in Private Information Retrieval: Formal Definition and Efficient Constructions

Reo Eriguchi[1*], Kaoru Kurosawa[1,2,3], and Koji Nuida[1,4]

[1]National Institute of Advanced Industrial Science and Technology, Tokyo, Japan
[2]Research and Development Initiative, Chuo University, Tokyo, Japan
[3]ZenmuTech Inc., Tokyo, Japan
[4]Institute of Mathematics for Industry (IMI), Kyushu University, Fukuoka, Japan
[*]Corresponding author: eriguchi-reo@aist.go.jp

**Abstract**

Multi-server Private Information Retrieval (PIR) is a cryptographic primitive that allows a client to retrieve an item of a database shared by multiple servers without revealing the index. This paper addresses the problem of error correction in multi-server PIR, enabling the client to obtain the item even when some servers provide incorrect responses. In a dishonest-majority setting where the majority of servers may introduce errors, it is known that the client can no longer uniquely determine the correct value. Previous approaches in this setting have typically settled for relaxed guarantees that the client can only reject incorrect values. However, these guarantees are substantially weak, as they only indicate the presence of errors without providing any information about the desired item. In this paper, we explore a more natural alternative called list-decodable PIR, which ensures that the client receives a small list of candidate values one of which is correct. We provide the first formal definition of list-decodable PIR and study its basic properties including a fundamental lower bound on the number of servers and the difficulty of simulation-based security definitions. We propose generic constructions of list-decodable PIR from any semi-honest PIR protocols, each offering different trade-offs. Our constructions improve upon the communication complexity of the only previously known protocol satisfying our definition. Furthermore, they achieve communication complexity comparable to that of the currently best known semi-honest PIR protocols.

## 1 Introduction

Private Information Retrieval (PIR) [11] is a cryptographic primitive that enables a client to retrieve a data item $D_x$ of his choice from a database $D = (D_1, \ldots, D_N)$ while hiding the index $x \in [N] := \{1, \ldots, N\}$ from $m$ servers sharing $D$. A trivial solution is to send the entire database to the client but it results in high communication complexity proportional to the database size $N$. A series of works [1, 4, 5, 10, 11, 19, 20, 30, 43] starting with Chor et al. [11] showed how to construct PIR protocols with non-trivial communication complexity $o(N)$ assuming $m \geq 2$ servers. This paper mainly focuses on information-theoretically secure multi-server PIR protocols. Note that computationally secure single-server protocols (e.g., [34, 35, 13]) have also been studied but they necessarily rely on relatively strong assumptions [16].

**Maliciously Secure PIR in the Dishonest-majority Setting.** As more servers are involved, there is a higher possibility that servers return incorrect answers. For example, servers may be malicious or faulty, or compute answers from an out-of-date database. Beimel and Stahl [6] introduced the notion of error-correcting PIR (also known as Byzantine-robust PIR), in which a client can uniquely determine a correct value even if servers' answers contain errors. Subsequently, efficient error-correcting PIR protocols were proposed [6, 33, 23, 46, 24]. However, it was shown in [6, 23] that error-correcting PIR is possible only if the number of errors is less than half the number of servers.

In this work, we consider the following natural question: Is there any alternative notion to error-correcting PIR in the *dishonest-majority* setting where the majority of servers may return incorrect answers? Previous

approaches in this setting have typically settled for relaxed guarantees that the client can only reject incorrect values [22, 45, 12]. However, these substantially weak guarantees only indicate the presence of errors without providing any information on the desired item $D_x$. To the best of our knowledge, the only exception is a PIR protocol by Goldberg [26], which introduced the idea of list decoding, a natural alternative to unique decoding in coding theory [40], into the context of PIR. In this protocol, a client not only detects the presence of errors but obtains a small list of candidate values one of which is correct. In [26], however, the notion of "list-decodable PIR" was not formally defined and their protocol required $\Omega(\sqrt{N})$ communication, which could be improved a lot in light of the state-of-the-art semi-honest protocols with communication complexity $N^{o(1)}$ [19].

**Usefulness of List-decodable PIR.** The concept of list-decodable PIR is not only theoretically intriguing but also offers two key practical advantages. First, the previous approaches in [22, 45, 12] are vulnerable to *denial-of-service* (DoS) attacks, where even a single corrupted server can repeatedly cause the client to fail, preventing him from obtaining any meaningful result, since these protocols only indicate the presence of errors. In contrast, list-decodable PIR enables the client to narrow down the initial set of many possible values into a constant number of candidates after a single protocol execution. Furthermore, when databases are authenticated using signatures or vector commitments, the client can efficiently filter out all but one of the candidate values, as we will formally demonstrate. This is inherently impossible in [22, 45, 12] since the client's output provides no information on the correct value.

Second, as pointed out in [12], another typical attack in the malicious setting is a *selective-failure attack*, where corrupted servers manipulate a database and infer his query by observing whether the client accepts or rejects the result. Consequently, the client needs to "grin and bear it" as his privacy will be compromised if he complains about the result. While *authenticated* PIR, introduced in [12] as a counter measure, allows the client to safely abort, the existing constructions have failed to achieve information-theoretic security and remain "ad-hoc", relying on protocol-specific modifications to particular PIR protocols. On the other hand, list-decodable PIR natively prevents selective-failure attacks since the client always receives valid results and never aborts the protocol, and can provide an alternative approach to mitigating these attacks. Notably, we will later show generic and information-theoretic constructions of list-decodable PIR protocols from any semi-honest PIR protocols.

## 1.1 Our Results

In this paper, we initiate a formal and in-depth study of list-decodable PIR (LD-PIR, for short). We discuss its basic properties including a fundamental lower bound on the number of servers and the difficulty of simulation-based security definitions. We propose efficient constructions of LD-PIR protocols with different tradeoffs. Our constructions not only improve the communication complexity $O(\sqrt{N})$ of the previous protocol in [26] but some of them even achieve communication complexity $N^{o(1)}$, which asymptotically matches the communication complexity of state-of-the-art semi-honest PIR protocols [19, 20].

**Formalization and Basic Properties.** We define PIR protocols as $(b, L)$-*list-decodable* if a client can compute a list $\mathcal{L}$ of candidate values, from servers' answers containing $b$ errors, such that the desired item $D_x$ is in $\mathcal{L}$ and the size of $\mathcal{L}$ does not exceed $L$. The condition on the list size is necessary because, without it, a client could simply output the set of all possible values as $\mathcal{L}$. Note that the above definition includes the definition of error-correcting PIR as a special case of $L = 1$. We then show that the number $m$ of servers in $(b, L)$-LD-PIR must exceed a lower bound $m > b(L + 1)/L$, which we call the *list decoding boun*d.

We also discuss the difficulty of the simulation-based security of LD-PIR in the dishonest-majority setting. The crux of the difficulty is that the simulation-based security assumes a central adversary who colludes with the majority of $b > m/2$ servers and tries to break the query privacy and the output correctness simultaneously. We show that under certain parameters, an efficient protocol that is secure against such adversaries implies an efficient single-server PIR protocol, which is impossible in the information-theoretic setting [11, 14]. Our game-based definition gets around the difficulty by specifying privacy and list-decoding requirements separately, in contrast to the simulation-based one where these requirements are defined in a unified way. We note that this corruption model has been adopted in the literature [6, 23, 26, 46].

**Efficient Constructions.** As a warm-up, we show a naive LD-PIR protocol based on polynomial interpolation. This is derived by applying the existing list-decoding algorithms for linear codes [28, 31, 32] to the PIR protocol of [42]. While this naive protocol improves upon the only known LD-PIR protocol [26], its

Table 1: Comparison of list-decodable PIR protocols

| Protocol | Number of servers | Communication | Remark |
|---|---|---|---|
| [26] | $m > \frac{L+1}{L}b + \Omega(Lt)$ | $\widetilde{O}(N^{1/2}m)$ | |
| Proposition 5.1 | $m > \frac{L+1}{L}b + \Omega(Ldt)$ | $\widetilde{O}(N^{1/d}md)$ | |
| Corollary 6.2 + [19] | $m > \max\left\{\frac{L+1}{L}b, b + 2^t\right\}$ | $N^{o_t(1)}m^{O(2^t)}$ | $\binom{m}{b}$ computation |
| Corollary 7.2 + [19] | $m > \max\left\{\frac{L+1}{L}b, b + 2^{t+1}\right\}$ | $N^{o_t(1)}m^2b2^{O(2^t)}$ | two-round |
| Corollary 7.6 + [19] | $m > \max\left\{\frac{L+1}{L}b, b + 2^t\right\}$ | $N^{o_t(1)}m^2b2^{O(2^t)}$ | local error model |
| Corollary 7.9 + [19] | $m > \max\left\{\frac{L+1}{L}b, b + 2^t\right\}$ | $N^{o_t(1)}m^3b2^{O(2^t)}$ | $\binom{m}{b}$ computation |

$N$ is the database size, $m$ is the number of servers, $b$ is the number of errors, $L$ is the list size, $t$ is the collusion size, and $d$ is a parameter. $o_t(1)$ is a function that converges to 0 when $N \to \infty$ and $t$ is fixed. The "local error model" means that each corrupted server introduces errors solely based on its own view, rather than on the joint view of all corrupted servers. See Section 8 and Appendix F for other instantiations.

communication complexity is not comparable to the state-of-the-art semi-honest protocols with $N^{o(1)}$ communication [19, 20]. Furthermore, the number of servers is not optimal as it is larger than the list decoding bound by an additive factor of $\Omega(Lt)$, where $t$ is the collusion size.

We then propose two sets of generic constructions of LD-PIR from *any* semi-honest PIR including [19, 20]. As a result, we obtain concrete LD-PIR protocols with $N^{o(1)}$ communication whose numbers of servers approach the list decoding bound.

The first construction follows a simple approach of performing an exhaustive search to identify error locations. This construction results in communication overheads that are not bounded by a fixed polynomial in the number $m$ of servers. Nonetheless, it is useful in real-world settings involving a small number of servers. From a theoretical perspective, we further propose a second set of constructions that achieve communication overheads bounded by a fixed polynomial in $m$. These constructions leverage different cut-and-choose techniques to catch corrupted servers and offer the following tradeoffs:

- The first variant achieves polynomially bounded communication and computation overheads, but requires two rounds of interaction.
- The second variant is one-round but can only correct errors introduced by non-colluding servers.
- The third variant is also one-round but incurs an exponential computation overhead of $\binom{m}{b}$.

It remains an open question whether one-round LD-PIR can be constructed from regular PIR with $\mathsf{poly}(m)$ communication and computation overheads.

By instantiating these constructions with a state-of-the-art protocol of [19], we obtain concrete LD-PIR protocols with communication complexity $N^{o(1)}$ as summarized in Table 1. Our protocols offer the following advantages:

- The number of servers is asymptotically optimal, meeting the list decoding bound $m > b(L+1)/L$ when the number of errors is sufficiently large, e.g., $b \geq L2^t$ in the best case.
- The communication complexity matches that of [19] up to a factor depending only on $b$, $t$ and $m$. That is, it is asymptotically equal to that of [19] with respect to the database size $N$.

We also provide instantiations based on other incomparable protocols from [20, 42], and compare their performance under concrete parameter settings. For details, see Section 8 and Appendix F.

**Unique Decoding with a Trusted Setup.** As mentioned above, unique decoding (i.e., $(b, 1)$-list-decoding) is impossible in the dishonest majority setting. We show that if we assume a trusted party who authenticates the contents of a database, list-decodable PIR can be efficiently upgraded into $(b, 1)$-LD-PIR even in the dishonest majority setting. This is in contrast to the prior PIR protocols with weak security guarantees [22, 45, 12], which cannot be upgraded in the same manner since corrupted servers can force an abort without any consequences regardless of whether the database is authenticated or not. Our transformations additionally use a digital signature scheme when the database is static, and use a vector commitment scheme [9, 41] when the database is possibly updated. Computational security is considered only in this section. We note that a uniquely decodable PIR protocol with a trusted setup is also obtained from a computationally secure 1-server PIR protocol. The advantage of using LD-PIR is that resulting protocols can be based on weaker cryptographic primitives not known to imply 1-server PIR such as one-way functions and collision-resistant hash functions.

More details on our techniques will be given in Section 2.

### 1.2 Related Works

The problem of verifying the correctness of servers' answers has been explored both in the multi-server setting [44, 22, 45, 12] and in the single-server setting [7, 12, 8, 17, 25]. As mentioned earlier, the multi-server protocols in [44, 22, 45, 12] offer only weak guarantees, allowing a client to reject incorrect values. In the single-server setting, on the other hand, the goals of [7, 12, 8, 17, 25] are to guarantee that a server's response is consistent with the committed database or a set of databases that meet certain predicates.

To perform error correction in the dishonest-majority setting, Goldberg [26] proposed a PIR protocol in which a client runs a well-known list decoding algorithm for Reed-Solomon codes [27] viewing a tuple of servers' answers as a codeword. Their protocol satisfies our definition of $(b, L)$-LD-PIR if $m > b(L+1)/L + \Omega(Lt)$. In the original work, they set the list size as $L = \Theta(\sqrt{m/t})$ to maximize the number of correctable errors. However, the communication complexity of their protocol is at least $\Omega(\sqrt{N})$, which is significantly larger than that of the state-of-the-art semi-honest protocol (e.g., [19]). Devet et al. [15] also considered error correction in the dishonest-majority setting but they assumed that injected errors are random and independent of a client's queries. In contrast, our model allows corrupted servers to inject even adversarial errors depending on a client's queries.

### 1.3 Open Problems

To argue the difficulty of simulation-based security of LD-PIR, we assume that the protocol has the minimum list size, that is, $L$ is the smallest integers satisfying the list decoding bound. It is unclear whether our impossibility results extend to settings where $L$ is allowed to be larger than the minimum.

Another main open problem is whether we can obtain a one-round LD-PIR protocol whose communication and client-side computation complexity are $N^{o(1)}\mathsf{poly}(m)$ for some fixed polynomial $\mathsf{poly}(m)$. Our current constructions with comparable complexity either require two rounds of interaction (Theorem 7.1) or can only correct errors introduced by non-colluding servers (Theorem 7.5).

## 2 Technical Overview

In this section, we provide high level ideas behind our techniques. We will give more detailed descriptions and security proofs in the subsequent sections.

### 2.1 Formalization and Basic Properties

**Formalization.** PIR protocols allow a client with input $x \in [N]$ to obtain a value $D_x \in \mathcal{X}$ by communicating with multiple servers holding a database $D = (D_1, \ldots, D_N) \in \mathcal{X}^N$. The notion of *b-error-correcting* PIR (also known as Byzantine-robust PIR) [6] guarantees that the client can uniquely determine a correct value even if the servers' answers contain $b$ adversarial errors. However, it is known that error-correcting PIR is possible only if the majority of servers return correct answers, i.e., $m > 2b$. To accommodate more errors,

we introduce the notion of *list-decodable PIR* (LD-PIR, for short), which guarantees that a client obtains a list $\mathcal{L}$ of candidate values one of which is correct, meaning $D_x \in \mathcal{L}$. More formally, we provide a game-based definition of $(b, L)$-LD-PIR where an adversary corrupting $b$ servers wins if a client outputs a list $\mathcal{L} \subseteq \mathcal{X}$ such that $D_x \notin \mathcal{L}$ or the size of $\mathcal{L}$ exceeds $L$. The condition on the list size is necessary because, without it, the requirement is trivially satisfied by outputting $\mathcal{L} = \mathcal{X}$. Note that $(b, 1)$-LD-PIR is equivalent to $b$-error-correcting PIR. As we will show later, $(b, L)$-LD-PIR for $L \geq 2$ allows us to handle a strictly larger number of errors than $b$-error-correcting PIR does.

**List Decoding Bound.** First, we show a fundamental lower bound $m > b(L+1)/L$ on the number $m$ of servers required for $(b, L)$-LD-PIR, which we call the *list decoding bound*. Equivalently, the maximum number of correctable errors in $m$-server LD-PIR is less than $mL/(L+1)$. To this end, we here assume that $L = 2$ for simplicity. Let $H$ be the set of all honest servers. Note that $|H| = m - b$. If $m \leq b(L+1)/L = 3b/2$, then $b \geq 2(m-b)$ and thus the set of all corrupted servers can be partitioned into two subsets $S$ and $T$ each containing at least $m - b$ servers. Let $x = 1$ be a client's index and $D = (D_1, \ldots, D_N)$ be the original database. Now, consider the following attack scenario:

- The servers in $S$ compute responses to queries using a database $D' = (D'_1, \ldots, D'_N)$;
- The servers in $T$ compute responses to queries using a database $D'' = (D''_1, \ldots, D''_N)$,

where for each $i \in [N]$, $D_i$, $D'_i$ and $D''_i$ are pairwise distinct values. If a protocol is $(b, L)$-list-decodable, a list $\mathcal{L}$ returned by a client must include $D_1$. On the other hand, the attack described above can also be interpreted as a scenario where servers in $S$ are honest and those in $H \cup T$ are corrupted. From this point of view, the list $\mathcal{L}$ should also contain the first entry $D'_1$ of $D'$. Similarly, $\mathcal{L}$ should contain the first entry $D''_1$ of $D''$. Since any of three databases $D$, $D'$ and $D''$ is possible from the client's point of view, he has no way to narrow down the list size into $L = 2$.

Conversely, if the list decoding bound is satisfied, we can easily construct a $(b, L)$-LD-PIR protocol: A client downloads a whole database from each of $m$ servers and outputs the $L$ most frequent values appearing at the $x$-th position. From the assumption $m > b(L+1)/L$, the correct value $D_x$ is necessarily included in the $L$ most frequent values, which implies that the protocol is $(b, L)$-list-decodable.

If $L \geq 2$, then $mL/(L+1) > m/2$ and the number of correctable errors can be larger than that of $b$-error-correcting PIR. We note that, however, the above protocol has communication complexity $\Omega(N)$ and is not efficient. In Sections 2.2, 2.3 and 2.4, we will give different constructions of $(b, L)$-LD-PIR with communication complexity $o(N)$.

**LD-PIR Implies PIR with Fewer Servers.** Due to the list decoding bound, the list size $L$ must exceed $b/(m-b)$ for given values of $b, m$. We say that an LD-PIR protocol has the minimum list size if $L$ is the smallest integer satisfying $L > b/(m-b)$. We show that an $m$-server $(b, L)$-LD-PIR protocol $\Pi$ with the minimum list size implies a regular PIR protocol $\Pi'$ with similar communication complexity whose number of servers is $m - b$. As a corollary, in the dishonest-majority setting $b \geq m/2$, we can rule out the possibility of efficient $b$-private $(b, L)$-LD-PIR with the minimum list size. Indeed, it would imply $(m-b)$-private $(m-b)$-server PIR and hence efficient single-server PIR, which is impossible in the information-theoretic setting [11, 14].

To this end, we here assume that $L = 2$ for simplicity. Let $\mathsf{S}_1, \ldots, \mathsf{S}_{m-b}$ be $m - b$ servers involved in $\Pi'$ and $\mathsf{T}_1, \ldots, \mathsf{T}_m$ be $m$ virtual servers involved in $\Pi$. The fact that $L = 2$ is the minimum list size means that $b/(m-b) \geq 1$. Then, $m \geq 2(m-b)$ and the set $\{\mathsf{T}_1, \ldots, \mathsf{T}_m\}$ of virtual servers is partitioned into two subsets $H_1, H_2$ with $|H_1| = m - b$ and $|H_2| \geq m - b$. Our approach is to let each $\mathsf{S}_i$ in $\Pi'$ play the role of a virtual server $\mathsf{T}_i \in H_1$ in $\Pi$ and a client emulates the other virtual servers in $H_2$. Specifically, the client holding a retrieval index $x$ chooses a random element $y$ from $\mathcal{X}$ and constructs a dummy database $D'$ by setting $D'_x = y$ and $D'_i = 0$ for all $i \neq x$. He emulates an answer of $\mathsf{T}_i \in H_2$ based on the database $D'$. From a similar argument in the previous section, the list-decoding property ensures that both $D_x$ and $D'_x$ should be contained in a list $\mathcal{L}$ and $|\mathcal{L}| \leq L = 2$, which implies $\mathcal{L} = \{D_x, y\}$. Then, the client can deduce the correct value $D_x$ as $y \neq D_x$ except with probability $1/|\mathcal{X}|$ (the failure probability can be made negligible by parallel executions).

As an interesting application of the above implication, we can discuss the difficulty of the simulation-based security of LD-PIR in the dishonest-majority setting. The crux of the difficulty is that the simulation-based security specifies privacy and list-decoding properties in a unified way. That is, there is a central adversary

who colludes with the majority of $b > m/2$ servers and tries to break the privacy of a client's index $x$ and the correctness of his output list $\mathcal{L}$ simultaneously. This means that the protocol should be $b$-private and $(b, L)$-list-decodable. However, we have shown that such a protocol cannot be realized efficiently. Our game-based definition gets around the difficulty by separating the number $b$ of errors from the privacy threshold $t$. That is, the size of each collusion of servers trying to learn a client's index $x$ should be at most $t$ while the total number $b$ of errors is allowed to be larger than $t$. See Section 4.4 for details.

Note that the above transformation assumes that the list size $L$ is minimal. Therefore, it remains a possibility that a meaningful simulation-based security notion for LD-PIR can be defined when $L$ is allowed to be larger than the minimum. For instance, while we rule out such a notion for $(b, 3)$-LD-PIR in the regime $2m/3 \leq b < 3m/4$, our result does not apply to the regime $m/2 \leq b < 2m/3$ where $L = 3$ is not the minimum.

## 2.2 Naive LD-PIR Based on Polynomial Interpolation

As a warm-up, we show a simple LD-PIR protocol with sublinear communication complexity $o(N)$. This protocol is derived by combining Woodruff-Yekhanin's PIR protocol [42] and a list-decoding algorithm for multiplicity codes [28, 31, 32].

Specifically, in the output phase of the protocol of [42], a client obtains $\boldsymbol{v} = (f(i), f'(i))_{i\in[m]}$ for a polynomial $f(X)$ with $f(0) = D_x$, where we denote the derivative of $f(X)$ at point $X = \alpha$ by $f'(\alpha)$. The client recovers $f$ from $\boldsymbol{v}$ via an interpolation formula and outputs $f(0) = D_x$. If servers inject errors, the client obtains a vector $\boldsymbol{c} = \boldsymbol{v} + \boldsymbol{e}$, where $\boldsymbol{e} = (e_{i0}, e_{i1})_{i\in[m]}$ is an error vector such that $e_{i0} = e_{i1} = 0$ for $i \in H$ ($H$ is the set of honest servers). A key observation is that $\boldsymbol{v}$ forms a codeword in a (univariate) multiplicity code, which is a variant of Reed-Solomon codes augmented with derivatives. We can then apply the existing list-decoding algorithms for this code [28, 31, 32] and obtain a list of candidate values of $\boldsymbol{v}$. Our analysis shows that the resulting protocol is $(b, L)$-LD-PIR with communication complexity $O(N^{1/d})$ if $m \geq b(L + 1)/L + \Omega(Ldt)$ for a parameter $d$.

Note that it seems impossible to directly apply list-decoding algorithms based on polynomial interpolation to the currently best known PIR protocols [19, 20], as the interpolating polynomials have high degrees relative to the number of servers. Moreover, since the privacy amplification in [2] does not preserve algebraic structures, such list-decoding algorithms do not seem effective for $t$-private protocols obtained by applying [2] to the protocols [19, 20].

## 2.3 LD-PIR Based on an Exhaustive Search

We show a generic construction of an $m$-server LD-PIR protocol that can be applied *any* $k$-server PIR protocol including [19, 20]. The communication complexity of the resulting LD-PIR protocol is $\binom{m}{k}$ times larger than that of an initial PIR protocol. Our approach is simple: Let $k \leq m - b$ be the number of honest servers. A client executes independent instances of a $k$-server PIR protocol in parallel with all $k$-sized subsets of servers. The privacy is inherited from the underlying $k$-server protocol. Since there are $m - b$ honest servers, at least one instance of the $k$-server protocol is executed with a set consisting of honest servers only, which ensures that a correct value $D_x$ must appear in the candidate outputs of the $\binom{m}{k}$ instances. A technical difficulty lies in that since $\binom{m}{k} > L$ in general, the client needs to remove as many incorrect values as possible and narrow down the candidates to $L$.

To this end, we use an *error-detecting* PIR protocol [22, 23] instead of a regular PIR protocol. Specifically, an error-detecting PIR protocol is not capable of correcting errors but allows a client to detect malicious behavior of servers, in which case he outputs a special symbol $\perp$, as long as at least one server is honest. It is known that any $k$-server PIR protocol can be efficiently transformed into an error-detecting $k$-server PIR protocol with communication overhead of $\widetilde{O}(k^2)$ [23].

We here consider the case of $k = m - b$ for simplicity. Let $y_S$ be an output of an error-detecting $k$-server PIR protocol executed with a subset $S$ of $k$ servers. First, the client discards invalid outputs such that $y_S = \perp$ and adds only valid outputs $y_S \neq \perp$ to a list $\mathcal{L}'$. Clearly, $D_x \in \mathcal{L}'$ since $y_H = D_x$, where $H$ is the set of honest servers. To construct a final list $\mathcal{L}$ from $\mathcal{L}'$, we observe that if $S$ contains at least one honest server, the client outputs either the correct value $D_x$ or $\perp$ since he does not accept an incorrect value $y \in \mathcal{X} \setminus \{D_x\}$. Since invalid outputs $y_S = \perp$ are removed at the beginning, it is guaranteed that $y_S$ included in $\mathcal{L}'$ is equal

to $D_x$ if $S \cap H \neq \emptyset$. Therefore, we have that

$$\mathcal{L}' = \underbrace{\{y_S : S \cap H \neq \emptyset\}}_{=\{D_x\}} \cup \{y_S : S \cap H = \emptyset\}.$$

Based on this observation, we set $\mathcal{L}$ as a maximal subset of $\mathcal{L}'$ of the form

$$\mathcal{L} = \{y_{S_1}, \ldots, y_{S_\ell} : S_1, \ldots, S_\ell \text{ are pairwise disjoint}\}.$$

Note that such a subset $\mathcal{L}$ can be efficiently found in a greedy way. Due to the maximality, it must hold that $S_i \cap H \neq \emptyset$ for some $S_i$ and $D_x = y_{S_i} \in \mathcal{L}$. Furthermore, if $m > b(L+1)/L$, then $(L+1)k > m$ and there are no pairwise disjoint $k$-sized subsets $S_1, \ldots, S_\ell$ such that $\ell \geq L+1$. Thus, we have that $|\mathcal{L}| \leq L$ and the protocol is $(b, L)$-list-decodable.

If we instantiate the above generic construction with the PIR protocols [19, 20], then we can obtain LD-PIR protocols whose communication complexity is $N^{o(1)}$ and numbers of servers approach the list decoding bound. Note that, however, the communication overhead is not a fixed polynomial in $m$. Hence, this construction works efficiently only if the number $m$ of servers is small.

## 2.4 LD-PIR with $\mathsf{poly}(m)$ Communication Overheads

We show three generic constructions of LD-PIR from regular PIR whose overheads in communication complexity are bounded by fixed polynomials in the number $m$ of servers. There are different tradeoffs:

- The first one is a two-round construction.
- The second one is one-round but can list-decode "local" errors only.
- The third one is one-round but incurs a computation overhead of $\binom{m}{b}$.

Ideally, if a client finds the set of honest servers, then he can obtain a correct value $D_x$ from answers returned by these honest servers. These constructions are based on different cut-and-choose approaches to catch corrupted servers.

**Two-round Construction.** Let $\Pi$ be an $m$-server PIR protocol. We start with the following naive way: In the first round, the client receives answers from $m$ servers running $\Pi$ as usual; In the second round, he asks a special common query (specified below) to all servers. He then decomposes the set of $m$ servers into several groups $C_1, \ldots, C_h$ such that servers in each $C_i$ return the same reply to the common query. Finally, he computes a list $\mathcal{L}$ of $h$ candidate values each of which is computed from a set of answers returned by servers in each $C_i$ in the first round. Since all honest servers return the same reply to the common query, the set $H$ of honest servers is included in one of the $C_i$'s and the correct value $D_x$ is included in $\mathcal{L}$. To guarantee that the list size is at most $L$, we remove groups $C_i$ of size less than $m-b$ in the first place since $|H| = m-b$. If $m > b(L+1)/L$, then we can see that $m < (L+1)(m-b)$. On the other hand, it holds that $m \geq |C_1| + \cdots + |C_h| \geq h(m-b)$, which ensure that $|\mathcal{L}| \leq L$. However, there is an obvious attack that a corrupted server behaves honestly in the second round to be included in the same group as honest servers, but behaves maliciously in the first round to cause errors in the output phase.

Our countermeasure is to choose a target server $\mathsf{S}_i$ uniformly at random and ask "What is $\mathsf{S}_i$'s correct answer to the query in the first round?" to all servers other than $\mathsf{S}_i$. It leaks one additional query in $\Pi$ and decreases the privacy level from $t$ to $t-1$. In the output phase, the client forms groups of servers who sent the same reply to the above query, where we fix the reply of $\mathsf{S}_i$ to its answer in the first round. If a corrupted server is chosen as a target server $\mathsf{S}_i$, this technique successfully prevents the above attack. This is because if the corrupted server returns an incorrect answer in the first round, it is put into a different group from $H$ and its incorrect answer cannot cause errors. Although there remains a possibility that another corrupted server not chosen as a target server may cause errors, we can prevent it by running sufficiently many independent instances in parallel so that any corrupted server is eventually caught. This construction only incurs $\mathsf{poly}(m)$ overheads in communication and computation complexity.

Note that $\Pi$ needs to satisfy a robustness property for a client to compute an output from a subset of answers. Specifically, we require $\Pi$ to be *$k$-robust* [6] for some $k \leq m-b$, which guarantees that a client

can reconstruct a correct value from any $m-b$ out of $m$ honestly generated answers. We can use efficient compilers from regular PIR to robust PIR in [6] and plug the compiled protocol into the above construction.

**One-round Construction in the Local Error Model.** Next, we reduce the number of rounds in the previous construction by letting the client send servers first-round and second-round queries simultaneously. More precisely, the client generates *computation queries* $(q_i)_{i\in[m]}$ and *verification queries* $(r_i)_{i\in[m]}$ independently. Each server $\mathsf{S}_i$ is asked to return its own answer to $q_i$ as well as the answer to $r_\ell$ that a randomly chosen server $\mathsf{S}_\ell$ should return. A list-decoding procedure proceeds similarly to the previous construction. The client partitions the set of servers into groups based on the answers to $r_\ell$'s and for each group, he computes an output from their answers to $q_i$'s. This construction can be built upon $t$-private PIR protocols instead of $(t+1)$-private ones since the verification queries $r_\ell$ are generated independently of the computation queries $q_i$.

An important distinction is that the target server $\mathsf{S}_\ell$ must not be able to determine whether a given query will be used for output computation or for answer verification. Otherwise, $\mathsf{S}_\ell$ could cause errors by responding honestly to verification queries while providing incorrect answers to computation queries. To prevent this attack, we randomize the order of verification and computation queries. Then, $\mathsf{S}_\ell$ cannot distinguish between two types of queries better than a random guess.

We note that the resulting protocol is applicable only when errors are *local* in the sense that each corrupted server modifies its answer solely based on its own view, rather than on the joint view of all $b$ corrupted servers. This is because since if a target server sees queries for other corrupted servers, then it can trivially distinguish between verification and computation queries.

**One-round Construction in the Non-local Error Model.** We present an alternative one-round construction that is applicable when errors are computed based on the joint view of corrupted servers. A high-level idea is that a client $\mathsf{C}$ sends a verification query $r_\ell$ only to another randomly chosen server $\mathsf{S}_j$, instead of sending $r_\ell$ to all servers. Specifically, after generating computation queries $(q_i)_{i\in[m]}$ and verification queries $(r_i)_{i\in[m]}$, $\mathsf{C}$ chooses a pair $(\mathsf{S}_\ell, \mathsf{S}_j)$ of servers uniformly at random. He then sends $(q_j, r_\ell)$ to $\mathsf{S}_j$ and sends $(q_i, r_i)$ to other servers $\mathsf{S}_i$. Again, we randomize the order of computation and verification queries so that corrupted servers cannot distinguish. We note that a similar query design was used for error detection in PIR [23]. Our contribution lies in the following novel technique for list-decoding errors.

Our list decoding algorithm proceeds as follows. The client $\mathsf{C}$ constructs a graph $G$ on $m$ nodes each representing a server. An edge between nodes $\ell, j$ indicates that the responses from $\ell$ and $j$ do not conflict. More specifically, $G$ is initially set to a complete graph. For a chosen pair $(\ell, j)$ of servers, $\mathsf{C}$ checks if their answers to the verification query $r_j$ are equal. If their answers are not equal, then $\mathsf{C}$ removes an edge $(\ell, j)$ from $G$. $\mathsf{C}$ iterates this process for sufficiently many times.

If a chosen pair of servers behave honestly, then the verification always passes and the edge between them remains. In particular, the set $H$ of honest servers forms a *clique* of size $m-b$ in the graph $G$. Recall that a clique is a subset of vertices such that every pair of vertices in it are adjacent. Based on this observation, $\mathsf{C}$ finds any clique of size $m-b$ in $G$ and removes the nodes in it from $G$. He continues this procedure until no clique of size $m-b$ is found. Finally, for each of the obtained cliques, $\mathsf{C}$ computes an output from their answers to the computation queries. To let $\mathsf{C}$ output an incorrect list, a corrupted server must inject sufficiently many errors. Then, its answers are compared with those computed by honest servers with high probability, and the server is excluded from the clique of honest servers. This guarantees the list-decoding property.

A limitation of the construction is its computation overhead: in the output phase, $\mathsf{C}$ needs to find a clique of size $m-b$ in a graph with $m$ nodes. Since the problem of finding cliques is conjectured to have no fixed polynomial-time algorithm [18], the brute-force search remains the most practical approach, incurring a computational overhead of $\binom{m}{b}$.

## 2.5 Unique Decoding with a Trusted Setup

As shown in [6, 23], unique decoding (i.e., $(b,1)$-list-decoding) is impossible in the dishonest majority setting. We consider a special model of *PIR with a trusted setup*, where a trusted party authenticates a database $D$ in the setup, and we show that list-decodable PIR can be efficiently upgraded into $(b,1)$-list-decodable PIR in this model even when $b \geq m/2$.[1] Specifically, we present a $(b,1)$-list-decodable PIR protocol based on

[1] While the idea of introducing such a trusted party was used by [44], their PIR protocol does not guarantee unique decoding.

a signature scheme, in which the trusted party signs each content of $D$, distributes these signatures along with $D$ among servers, and sends a client a verification key. After the setup, the client and servers execute a list-decodable PIR protocol as usual except that servers compute answers based on the singed database instead of the original one. The client computes a list from servers' answers and verifies the validity of each item in the list. The resulting protocol achieves unique decoding since the list-decoding property guarantees that $D_x$ is included in the output list and the security of the signature scheme guarantees that any incorrect element is excluded in the verification phase.

Although the above protocol works well for a static database, when even a single element of the database is updated, the trusted party needs to conduct the setup again, which results in $O(N)$ communication between the trusted party and servers. We also propose a PIR protocol with a trusted setup that allows for efficient database updates, based on vector commitments [9, 41], which can be instantiated from collision-resistant hash functions.

Note that if a computationally secure 1-server PIR protocol is assumed, then we can obtain a $(b+1)$-server PIR protocol with a trusted setup achieving unique decoding against $b$ corrupted servers. A client just executes the 1-server protocol with each server and outputs a value that passes verification. The advantage of using LD-PIR is that resulting protocols can be based on weaker cryptographic primitives not known to imply 1-server PIR. In fact, according to the results of [16], it is supposedly impossible to obtain 1-server PIR from one-way functions, which are sufficient for digital signatures [36]. Furthermore, 1-server PIR is not known to be obtained from collision-resistant hash functions while the other direction is known [29].

# 3 Preliminaries

**Notations.** For a natural number $n \in \mathbb{N}$, we define $[n] = \{1, 2, \ldots, n\}$. Let $X, Y$ be sets. If $X \subseteq Y$, we define $Y \setminus X = \{y \in Y : y \notin X\}$ and simply denote it by $\overline{X}$ if $Y$ is clear from the context. We write $u \leftarrow\!\!\$\, X$ if $u$ is chosen uniformly at random from $X$. We denote the set of all $k$-sized subsets of $X$ by $\binom{X}{k}$. Let $\log x$ denote the base-2 logarithm of $x$ and $\ln x$ denote the base-e logarithm of $x$, where e is Napier's constant. We call $f : \mathbb{N} \ni \lambda \mapsto f(\lambda) \in \mathbb{R}$ a *negligible function* if for any $c > 0$, there exists $\lambda_0 \in \mathbb{N}$ such that $0 \leq f(\lambda) < \lambda^{-c}$ for any $\lambda > \lambda_0$. We call $f$ a *polynomial function* if there exists $c > 0$ such that $0 \leq f(\lambda) < \lambda^c$ for all $\lambda$.

Throughout the paper, we fix the following notations:

- $\lambda$ is a security parameter.
- $m$ is the number of servers.
- $\mathcal{X}^N$ is the set of all databases: A database $D = (D_1, \ldots, D_N)$ has $N$ components and each component $D_i$ takes a value from $\mathcal{X}$.
- $\mathsf{C}$ is a client and $x \in [N]$ denotes his query.
- $\mathsf{S}_i$ is the $i$-th server holding a copy of a database $D \in \mathcal{X}^N$.
- $t$ is the privacy threshold: at most $t$ servers may collude to infer $x$.
- $b$ is the number of errors: at most $b$ servers may return incorrect answers.
- $L$ is the size of a list outputted by the client.

Unless stated otherwise, we assume that $m$, $t$, $b$ and $L$ are polynomial functions in $\lambda$. The notation $\widetilde{O}$ hides a polylogarithmic factor in $\lambda$ and $m$.

## 3.1 Private Information Retrieval

We assume $m$ servers $\mathsf{S}_1, \ldots, \mathsf{S}_m$ who all hold a database $D = (D_1, \ldots, D_N) \in \mathcal{X}^n$ and an honest client $\mathsf{C}$ who holds a private index $x \in [N]$. The goal of a Private Information Retrieval (PIR) protocol is:

**Correctness.** The client learns a value $D_x$;

**Privacy.** The client keeps his index $x$ hidden from any collusion of $t$ servers.

Formally, we define a PIR protocol as follows. For simplicity, we here focus on one-round PIR protocols and defer the definition of general multi-round protocols to Appendix A.

**Definition 3.1.** *A* (1-round) $t$-private $m$-server PIR protocol *is a tuple of three polynomial-time algorithms* $\Pi = (\mathsf{Query}, \mathsf{Answer}, \mathsf{Output})$*, where:*

- $\mathsf{Query}(1^\lambda, x) \to ((\mathsf{que}_i)_{i\in[m]}, \mathsf{st})$*:* $\mathsf{Query}$ *is a randomized algorithm that takes an index* $x \in [N]$ *as input and outputs queries* $(\mathsf{que}_i)_{i\in[m]}$ *and a state* $\mathsf{st}$*, where we assume that the index* $i$ *is implicitly included in the description of* $\mathsf{que}_i$*;*
- $\mathsf{Answer}(D, \mathsf{que}_i) \to \mathsf{ans}_i$*:* $\mathsf{Answer}$ *is a deterministic algorithm that takes a database* $D \in \mathcal{X}^N$ *and a query* $\mathsf{que}_i$ *as input and outputs an answer* $\mathsf{ans}_i$*;*
- $\mathsf{Output}(\mathsf{st}, (\mathsf{ans}_i)_{i\in[m]}) \to y$*:* $\mathsf{Output}$ *is a deterministic algorithm that takes a state* $\mathsf{st}$ *and answers* $(\mathsf{ans}_i)_{i\in[m]}$ *as input and outputs* $y \in \mathcal{X}$*;*

*such that* $\Pi$ *satisfies the following properties:*

**Correctness.** *For any* $\lambda \in \mathbb{N}$*, any* $D = (D_1, \ldots, D_N) \in \mathcal{X}^N$ *and any* $x \in [N]$*, it holds that*

$$\Pr\left[\mathsf{Output}(\mathsf{st}, (\mathsf{ans}_i)_{i\in[m]})) = D_x\right] = 1, \tag{1}$$

*where* $((\mathsf{que}_i)_{i\in[m]}, \mathsf{st}) \leftarrow \mathsf{Query}(1^\lambda, x)$ *and* $\mathsf{ans}_i \leftarrow \mathsf{Answer}(D, \mathsf{que}_i)$ *for all* $i \in [m]$*.*

**Privacy.** *For any stateful algorithm* $\mathcal{A}$*, the distributions of* $\mathsf{Priv}^0_{\Pi,\mathcal{A}}(\lambda)$ *and* $\mathsf{Priv}^1_{\Pi,\mathcal{A}}(\lambda)$ *are statistically indistinguishable, where for* $\alpha \in \{0,1\}$*,* $\mathsf{Priv}^\alpha_{\Pi,\mathcal{A}}(\lambda)$ *is the output* $\alpha'$ *of* $\mathcal{A}$ *in the following experiment* $\mathrm{EXPT}^{\mathrm{Priv},\alpha}_{\Pi,\mathcal{A}}(\lambda)$*:*

1. $\mathcal{A}(1^\lambda)$ *outputs* $(x_0, x_1, D, T)$*, where* $x_0, x_1 \in [N]$*,* $D \in \mathcal{X}^N$ *and* $T \subseteq [m]$ *is of size at most* $t$*.*
2. *Let* $((\mathsf{que}_i)_{i\in[m]}, \mathsf{st}) \leftarrow \mathsf{Query}(1^\lambda, x_\alpha)$ *and give* $(\mathsf{que}_i)_{i\in T}$ *to* $\mathcal{A}$*.*
3. $\mathcal{A}$ *outputs a bit* $\alpha' \in \{0,1\}$*.*

We omit a security parameter $1^\lambda$ from descriptions if it is clear from the context.

The *communication complexity* $\mathrm{Comm}(\Pi)$ of $\Pi$ is defined as the total bit-length of queries $(\mathsf{que}_i)_{i\in[m]}$ and answers $(\mathsf{ans}_i)_{i\in[m]}$ with worst-case inputs $D \in \mathcal{X}^N, x \in [N]$. The *computation complexity* $\mathrm{Comp}(\Pi)$ of $\Pi$ is defined as the total running time of $\mathsf{Query}$, $\mathsf{Answer}$, and $\mathsf{Output}$, with worst-case inputs $(D, x)$. We see the communication complexity as a primary efficiency measure and the computation complexity as a secondary measure.

We also define the notion of $\epsilon$-correct PIR as the one satisfying the requirements in Definition 3.1 except that the condition (1) is replaced with

$$\Pr\left[\mathsf{Output}(\mathsf{st}, (\mathsf{ans}_i)_{i\in[m]})) = D_x\right] = 1 - \epsilon.$$

Note that we use the above weaker notion of PIR only when showing the relation between list-decodable PIR and single-server PIR.

## 3.2 Error-detecting PIR

Error-detecting PIR, introduced by [22, 23], is a PIR protocol ensuring that a client detects malicious behavior of servers, in which case he outputs a special symbol $\perp$.

**Definition 3.2.** *We say that a PIR protocol* $\Pi = (\mathsf{Query}, \mathsf{Answer}, \mathsf{Output})$ *is* $b$-error-detecting *if* $\mathsf{Output}$ *is allowed to output a special symbol* $\perp \notin \mathcal{X}$ *and* $\Pi$ *satisfies the following requirements:*

**Error detection.** *There exists a negligible function* $\epsilon = \epsilon(\lambda)$ *such that for any* $\lambda \in \mathbb{N}$ *and any stateful algorithm* $\mathcal{A}$*, it holds that*

$$\Pr[\mathsf{ED}_{\Pi,\mathcal{A}}(\lambda) = 1] \le \epsilon,$$

*where* $\mathsf{ED}_{\Pi,\mathcal{A}}(\lambda)$ *is the output of the following experiment:*

1. $\mathcal{A}(1^\lambda)$ *outputs* $(x, D, T)$*, where* $D \in \mathcal{X}^N$*,* $x \in [N]$ *and* $T \subseteq [m]$ *is of size at most* $b$*.*
2. *Let* $((\mathsf{que}_i)_{i\in[m]}, \mathsf{st}) \leftarrow \mathsf{Query}(1^\lambda, x)$ *and give* $(\mathsf{que}_i)_{i\in T}$ *to* $\mathcal{A}$*.*
3. $\mathcal{A}$ *outputs* $(\mathsf{ans}_i)_{i\in T}$*. Let* $\mathsf{ans}_i \leftarrow \mathsf{Answer}(D, \mathsf{que}_i)$ *for* $i \notin T$*.*
4. *Let* $y \leftarrow \mathsf{Output}(\mathsf{st}, (\mathsf{ans}_i)_{i\in[m]})$ *where* $y \in \mathcal{X} \cup \{\perp\}$*. Return* 0 *if* $y \in \{D_x, \perp\}$*, and otherwise return* 1*.*

*We call the bound* $\epsilon$ *the* failure probability *of* $\Pi$*.*

There exists a compiler from regular PIR to error-detecting PIR in [23]: A $t$-private 1-round $k$-server PIR protocol $\Pi$ can be transformed into a $t$-private $(m-1)$-error-detecting 1-round $m$-server PIR protocol $\Pi'$ such that $\mathrm{Comm}(\Pi') = \mathrm{Comm}(\Pi) \cdot \widetilde{O}(m^2)$ and $\mathrm{Comp}(\Pi') = \mathrm{Comp}(\Pi) \cdot \widetilde{O}(m^2)$.

### 3.3 Robust PIR

$k$-Robust PIR, introduced by [6], is a PIR protocol ensuring that a client can recover a correct value $D_x$ from any $k$ out of $m$ honestly generated answers.

**Definition 3.3.** *Let* $\Pi = (\mathsf{Query}, \mathsf{Answer}, \mathsf{Output})$ *be an* $m$*-server PIR protocol. We say that* $\Pi$ *is* $k$-robust *if* $\mathsf{Output}$ *additionally takes a subset* $A \subseteq [m]$ *of size* $k$ *and* $\Pi$ *satisfies the following requirements:*

**Robustness.** *For any* $\lambda \in \mathbb{N}$*, any* $D = (D_1, \ldots, D_N) \in \mathcal{X}^N$ *and any* $x \in [N]$*, it holds that*

$$\Pr[\mathsf{Output}(A, \mathsf{st}, (\mathsf{ans}_i)_{i\in A})) = D_x] = 1,$$

*where* $((\mathsf{que}_i)_{i\in[m]}, \mathsf{st}) \leftarrow \mathsf{Query}(1^\lambda, x)$ *and* $\mathsf{ans}_i \leftarrow \mathsf{Answer}(D, \mathsf{que}_i)$ *for all* $i \in [m]$*.*

There exist two compilers from regular PIR to robust PIR in [6]: Let $\Pi$ be a 1-round $k$-server PIR protocol.

- $\Pi$ can be transformed into a 1-round $k$-robust $m$-server PIR protocol $\Pi'$ such that $\mathrm{Comm}(\Pi') = \mathrm{Comm}(\Pi) \cdot \widetilde{O}(ke^k m)$.
- For any $k'$ with $k'/\ln k' \geq k$ (i.e., $k' = \Omega(k \log k)$), $\Pi$ can be transformed into a 1-round $k'$-robust $m$-server PIR protocol $\Pi''$ such that $\mathrm{Comm}(\Pi'') = \mathrm{Comm}(\Pi) \cdot \widetilde{O}(km)$.

These compilers are incomparable as the first one achieves better robustness while the second one reduces a communication overhead depending on $k$.

## 4 List-decodable PIR

### 4.1 Formalization

We introduce a formal definition of *list-decodable PIR* (LD-PIR, for short), which guarantees that a client obtains a list $\mathcal{L}$ of candidate values one of which is correct even if the servers' answers contain $b$ errors. More formally, we say that a PIR protocol is $(b, L)$-list-decodable if an adversary $\mathcal{A}$ corrupting $b$ servers cannot let a client output a list $\mathcal{L}$ such that $D_x \notin \mathcal{L}$ or the size of $\mathcal{L}$ exceeds $L$. This notion is a generalization of error-correcting PIR (also known as Byzantine-robust PIR) [6], where $\mathcal{L}$ consists solely of the correct value, i.e., $L = 1$. As we will see later, by allowing the list size to be greater than one, LD-PIR can handle a larger number of errors compared to error-correcting PIR. For simplicity, we here present the definition of one-round LD-PIR and defer the general definition of multi-round one to Appendix A.

**Definition 4.1.** *Let* $\Pi = (\mathsf{Query}, \mathsf{Answer}, \mathsf{Output})$ *be an* $m$*-server PIR protocol. We say that* $\Pi$ *is* $(b, L)$-list-decodable *if* $\mathsf{Output}$ *is allowed to output a subset* $\mathcal{L} \subseteq \mathcal{X}$ *and* $\Pi$ *satisfies the following requirements:*

**List decoding.** *There exists a negligible function* $\epsilon = \epsilon(\lambda)$ *such that for any* $\lambda \in \mathbb{N}$ *and any stateful algorithm* $\mathcal{A}$*, it holds that*

$$\Pr[\mathsf{LD}_{\Pi,\mathcal{A}}(\lambda) = 1] \leq \epsilon,$$

*where* $\mathsf{LD}_{\Pi,\mathcal{A}}(\lambda)$ *is the output of the following experiment* $\mathrm{EXPT}^{\mathrm{LD}}_{\Pi,\mathcal{A}}(\lambda)$*:*

1. $\mathcal{A}(1^\lambda)$ *outputs* $(x, D, T)$*, where* $D \in \mathcal{X}^N$*,* $x \in [N]$ *and* $T \subseteq [m]$ *is of size at most* $b$*.*
2. *Let* $((\mathsf{que}_i)_{i\in[m]}, \mathsf{st}) \leftarrow \mathsf{Query}(1^\lambda, x)$ *and give* $(\mathsf{que}_i)_{i\in T}$ *to* $\mathcal{A}$*.*
3. $\mathcal{A}$ *outputs* $(\mathsf{ans}_i)_{i\in T}$*. Let* $\mathsf{ans}_i \leftarrow \mathsf{Answer}(D, \mathsf{que}_i)$ *for* $i \notin T$*.*
4. *Let* $\mathcal{L} \leftarrow \mathsf{Output}(\mathsf{st}, (\mathsf{ans}_i)_{i\in[m]})$*. Return* 0 *if*

$$D_x \in \mathcal{L} \text{ and } |\mathcal{L}| \le L, \tag{2}$$

*and otherwise return* 1*.*

*We call the bound* $\epsilon$ *the* failure probability *of* $\Pi$*.*

Note that if $L \ge |\mathcal{X}|$, then the list decoding property is trivially satisfied. Indeed, $\mathsf{Output}$ is allowed to output the set of all possible values, i.e., $\mathcal{L} = \mathcal{X}$, and the protocol is then $(b, L)$-list-decodable. From this point of view, we suppose throughout the paper that $L < |\mathcal{X}|$.

## 4.2 List Decoding Bound

We derive a fundamental lower bound on the number $m$ of servers in $(b, L)$-LD-PIR protocols.

**Theorem 4.2.** *If* $m \le b(L+1)/L$*, there exists no* $(b, L)$*-LD-PIR protocol.*

*Proof.* Assume otherwise that there exists an $\ell$-round $m$-server $(b, L)$-LD-PIR protocol $\Pi = (\mathsf{Query}, \mathsf{Answer}, \mathsf{Output})$. We can obtain from $\Pi$ a PIR protocol $\Pi' = (\mathsf{Query}', \mathsf{Answer}', \mathsf{Output}')$ such that $\mathsf{Query}' = \mathsf{Query}$, $\mathsf{Answer}' = \mathsf{Answer}$, and $\mathsf{Output}'$ is the same as $\mathsf{Output}$ except that whenever $\mathsf{Output}$ produces a list $\mathcal{L}$ whose size exceeds $L$, $\mathsf{Output}'$ clips a subset $\mathcal{L}'$ of size $L$ from $\mathcal{L}$ (e.g., the $L$ smallest value according to some order in $\mathcal{X}$). Note that $\Pi'$ is also $(b, L)$-list-decodable since whenever the success condition (2) is satisfied in $\Pi$, it is satisfied in $\Pi'$ as well. Furthermore, in $\Pi'$, a client never outputs a list of size more than $L$.

Since $(L+1)(m-b) \le m$, there exists a partition $(H_1, \ldots, H_{L+1})$ of the set of servers such that $|H_j| \ge m-b$ for all $j \in [L+1]$. Let $y_1, \ldots, y_{L+1} \in \mathcal{X}$ be distinct elements, whose existence is guaranteed by the assumption $L < |\mathcal{X}|$. Let $D^{(1)}, \ldots, D^{(L+1)} \in \mathcal{X}^N$ be $L+1$ databases such that the first element of $D^{(k)}$ is $y_j$, i.e., $D_1^{(k)} = y_k$ for all $k \in [L+1]$. For each $k \in [L+1]$, define an adversary $\mathcal{A}_k$ against the list decoding property of $\Pi'$ as follows:

1. $\mathcal{A}_k$ outputs $(x, D, T) = (1, D^{(k)}, \overline{H_k})$. Note that the size of $\overline{H_k}$ is at most $b$.
2. For each $h \ne k$, the servers in $H_h$ compute their responses by using $D^{(h)}$. Namely, for each round $j = 1, 2, \ldots, \ell$, given queries $(\mathsf{que}_i^{(j)})_{i\in\overline{H_k}}$ in round $j$, $\mathcal{A}_k$ computes $\mathsf{ans}_i^{(j)} \leftarrow \mathsf{Answer}'(D^{(h)}, \mathsf{que}_i^{(j)})$ for all $i \in \overline{H_k}$, where $h$ is such that $i \in H_h$, and returns $(\mathsf{ans}_i^{(j)})_{i\in\overline{H_k}}$.

Let $\mathcal{L}_k$ be the list returned in the experiment $\mathrm{EXPT}^{\mathrm{LD}}_{\Pi',\mathcal{A}_k}$ defined in Definition 4.1. Since $\Pi'$ is $(b, L)$-list-decodable, there exists a negligible function $\epsilon$ such that

$$\Pr[y_k \in \mathcal{L}_k] = \Pr[y_k \in \mathcal{L}_k \wedge |\mathcal{L}_k| \le L] = \Pr[\mathsf{LD}_{\Pi',\mathcal{A}_k} = 0] \ge 1 - \epsilon.$$

Note that all the experiments $\mathrm{EXPT}^{\mathrm{LD}}_{\Pi',\mathcal{A}_1}, \ldots, \mathrm{EXPT}^{\mathrm{LD}}_{\Pi',\mathcal{A}_{L+1}}$ are the same from a view point of the client. Thus, it holds that $\Pr[y_k \in \mathcal{L}_1] = \Pr[y_k \in \mathcal{L}_k] \ge 1 - \epsilon$ for all $k \in [L+1]$. Let $\mathsf{ALL}$ be the event that $y_k \in \mathcal{L}_1$ for all $k \in [L+1]$ in $\mathrm{EXPT}^{\mathrm{LD}}_{\Pi',\mathcal{A}_1}$. We then have that $\Pr[\mathsf{ALL}] \ge 1 - (L+1)\epsilon$. On the other hand, if $\mathsf{ALL}$ happens, then $|\mathcal{L}_1| \ge L+1$. Since the probability that the size of $\mathcal{L}_1$ exceeds $L$ is 0 by definition of $\Pi'$, we obtain that $1 - (L+1)\epsilon \le 0$ and hence $\epsilon \ge 1/(L+1)$, which contradicts to the fact that $\epsilon$ is negligible. □

Thus, a necessary condition for $m$-server $(b, L)$-LD-PIR is that the following *list decoding bound* is satisfied:

$$m > \frac{L+1}{L} b.$$

Equivalently, the list size $L$ must exceed $b/(m-b)$ for given values of $b$ and $m$. Accordingly, we say that an LD-PIR protocol has the *minimum list size* if $L$ is the smallest integer satisfying the list decoding bound, i.e.,

$$L = \left\lfloor \frac{b}{m-b} \right\rfloor + 1.$$

Note that the list decoding bound generalizes a necessary condition of error-correcting PIR that $m > 2b$ [6, 23], which is recovered by setting $L = 1$ in Theorem 4.2.

If we do not take into account efficiency, the converse of Theorem 4.2 is straightforward: If the list decoding bound is satisfied, then we can obtain an $m$-server $(b, L)$-LD-PIR protocol by running a single-server PIR protocol with each server and choosing the $L$ most frequent elements in the list of the $m$ outputs. See Appendix B for the formal statement and proof. However, in the information-theoretic setting, the resulting protocol must have communication complexity $\Omega(N)$ due to the lower bounds in [11, 14]. From this point of view, our main question is whether there exists an LD-PIR protocol whose communication complexity is sublinear in $N$ when the list decoding bound is satisfied.

## 4.3 LD-PIR Implies PIR with Fewer Servers

We show an implication from $m$-server $(b, L)$-LD-PIR with the minimum list size to regular PIR with $m - b$ servers.

**Theorem 4.3.** *Assume that $|\mathcal{X}| \geq 6(L-1)$. Let $\Pi$ be a $t$-private $\ell$-round $m$-server $(b, L)$-LD-PIR protocol with the minimum list size. Then, there exists a $t$-private $\mathsf{negl}(\lambda)$-correct $\ell$-round $(m-b)$-server PIR protocol $\Pi'$ such that $\mathrm{Comm}(\Pi') = \widetilde{O}(\mathrm{Comm}(\Pi))$.*

*Proof.* Let $\Pi = (\mathsf{Query}, \mathsf{Answer}, \mathsf{Output})$ be a given $t$-private $m$-server $(b, L)$-LD-PIR protocol. Let $\epsilon$ denote the failure probability of $\Pi$. Since $\epsilon$ is negligible, we may assume that $\epsilon \leq 1/(6L)$. Furthermore, by the same argument as the proof of Theorem 4.2, we may also assume that $\Pi$ always output a list of size at most $L$. We construct a $t$-private $\mathsf{negl}(\lambda)$-correct $(m-b)$-server PIR protocol $\Pi'$ from $\Pi$.

We call $m - b$ servers involved in $\Pi'$ *real servers* and $m$ servers involved in $\Pi$ *virtual servers*. Since $L = \lfloor b/(m-b) \rfloor + 1$, we have that $L \leq m/(m-b)$ and hence $m \geq L(m-b)$. There exists a partition $(H_1, H_2, \ldots, H_L)$ of the set of $m$ virtual servers such that $|H_1| = m - b$ and $|H_j| \geq m - b$ for $j \in \{2, \ldots, L\}$. Roughly speaking, in $\Pi'$, the $m - b$ real servers play the roles of virtual servers in $H_1$ and the client emulates the other virtual servers in $H_2 \cup \cdots \cup H_L$ based on dummy databases. Namely, the $t$-private $(m-b)$-server PIR protocol $\Pi'$ works as follows:

1. Let $x \in [N]$ be a client's index and $D \in \mathcal{X}^N$ be a database held by real servers.
2. The client chooses $L - 1$ elements $y_2, \ldots, y_L \leftarrow_{\$} \mathcal{X}$ uniformly at random and constructs $L - 1$ dummy databases $D^{(2)}, \ldots, D^{(L)} \in \mathcal{X}^N$ by defining each $D^{(j)} = (D_1^{(j)}, \ldots, D_N^{(j)})$ as

$$D_i^{(j)} = \begin{cases} y_j & \text{if } i = x, \\ y_{\mathrm{def}} & \text{otherwise,} \end{cases}$$

   where $y_{\mathrm{def}} \in \mathcal{X}$ is a default value.
3. The client emulates $\Pi$ by using the real servers of $\Pi'$ as follows. In each round of $\Pi$, he computes $m$ queries following the client-side procedures of $\Pi$. He then sends the real servers the first $m - b$ queries corresponding to virtual servers in $H_1$, and receives the corresponding answers. For each $j \in \{2, \ldots, L\}$, he locally simulates answers from virtual servers in $H_j$ assuming that a database is $D^{(j)}$.
4. The client computes a list $\mathcal{L} \subseteq \mathcal{X}$ by running $\mathsf{Output}$. He outputs any element $y \in \mathcal{L} \setminus \{y_2, \ldots, y_L\}$ if $\mathcal{L} \setminus \{y_2, \ldots, y_L\} \neq \emptyset$ and outputs the default value $y_{\mathrm{def}}$ otherwise.

The $t$-privacy of $\Pi'$ immediately follows from that of $\Pi$. Indeed, each real server in $\Pi'$ only sees one query for a virtual server in $\Pi$. Thus, the joint view of $t$ real servers in $\Pi'$ contains $t$ queries in $\Pi$, which are independent of $x$.

Next, we show the correctness of $\Pi'$. We will show that $y = D_x$ holds with probability at least $2/3$. Let $D^{(1)} := D$ and $y_1 := D_x$.

For each $k \in [L]$, consider an attack scenario against $\Pi$ in which $H_k$ is the set of honest servers and the true database is $D^{(k)}$ but for $h \in [L] \setminus \{k\}$, the corrupted servers in $H_h$ return answers based on $D^{(h)}$ instead of $D^{(k)}$. Formally, we define an adversary $\mathcal{A}_k$ against the $(b, L)$-list-decoding property of $\Pi$ as follows:

1. $\mathcal{A}_k$ outputs $(x, D, T) = (x, D^{(k)}, \overline{H_k})$.
2. For each $h \in [L] \setminus \{k\}$, the servers in $H_h$ compute their responses by using $D^{(h)}$. Namely, for each round $j \in [\ell]$, given quereis $(\mathsf{que}_i^{(j)})_{i \in \overline{H_k}}$, $\mathcal{A}_k$ computes $\mathsf{ans}_i^{(j)} \leftarrow \mathsf{Answer}(D^{(h)}, \mathsf{que}_i^{(j)})$ for all $i \in \overline{H_k}$, where $h$ is such that $i \in H_h$, outputs $(\mathsf{ans}_i^{(j)})_{i \in \overline{H_k}}$.

Let $\mathcal{L}_k$ be the list returned in the experiment $\mathrm{EXPT}^{\mathrm{LD}}_{\Pi,\mathcal{A}_k}$ defined in Definition A.2. Since $\Pi$ is $(b, L)$-LD-PIR with probability of failure $\epsilon$, and $|\overline{H_k}| \le b$, it holds that $\Pr[y_k \in \mathcal{L}_k] \ge 1 - \epsilon$. Note that all the experiments $\mathrm{EXPT}^{\mathrm{LD}}_{\Pi,\mathcal{A}_1}, \ldots, \mathrm{EXPT}^{\mathrm{LD}}_{\Pi,\mathcal{A}_L}$ are the same from a view point of the client. Thus, it holds that $\Pr[y_k \in \mathcal{L}_1] = \Pr[y_k \in \mathcal{L}_k] \ge 1 - \epsilon$ for all $k \in [L]$. Let $\mathsf{ALL}$ be the event that $y_k \in \mathcal{L}_1$ for all $k \in [L]$ in $\mathrm{EXPT}^{\mathrm{LD}}_{\Pi,\mathcal{A}_1}$. We then have that $\Pr[\mathsf{ALL}] \ge 1 - L\epsilon$.

Since the client's view during the execution of $\Pi'$ is the same as that in the experiment $\mathrm{EXPT}^{\mathrm{LD}}_{\Pi,\mathcal{A}_1}$, it follows from $|\mathcal{X}| \ge 6(L-1)$ that

$$\Pr[\{D_x, y_2, \ldots, y_L\} \subseteq \mathcal{L}] \ge 1 - L\epsilon \ge \frac{5}{6}.$$

The probability that the elements $D_x, y_2, \ldots, y_L$ are pairwise distinct, that is, $|\{D_x, y_2, \ldots, y_L\}| = L$, is

$$\frac{|\mathcal{X}| - 1}{|\mathcal{X}|} \cdot \frac{|\mathcal{X}| - 2}{|\mathcal{X}| - 1} \cdots \frac{|\mathcal{X}| - (L-1)}{|\mathcal{X}| - (L-2)} = \frac{|\mathcal{X}| - L + 1}{|\mathcal{X}|} \ge \frac{5}{6}$$

Therefore, since $\Pi$ always output a list of size at most $L$, the probability that $\mathcal{L} = \{D_x, y_2, \ldots, y_L\}$ holds is at least $1 - (1/6 + 1/6) = 2/3$. If this event happens, then $\mathcal{L} \setminus \{y_2, \ldots, y_L\} = \{D_x\}$ and the client outputs the correct result.

To amplify the success probability, the client just executes $\Pi'$ many times (say, $\kappa$ times) in parallel and outputs the majority of the $y$'s. Let $X_u$ be a Bernoulli random variable indicating success/failure of the $u$-th execution. We have seen that $p := \Pr[X_u = 1] \ge 2/3$ for all $u$. By Chernoff's bound, the failure probability of the final protocol is upper bounded by $\Pr[\sum_{u \in [\kappa]} X_u \le \kappa/2] \le \exp(-\Omega(\kappa))$. If we set $\kappa = (\log \lambda)^2$, then the probability is negligible in $\lambda$.

Finally, since a single execution of $\Pi'$ has lower communication complexity than $\Pi$, the communication complexity of the final protocol is $\kappa \cdot \mathrm{Comm}(\Pi') < (\log \lambda)^2 \cdot \mathrm{Comm}(\Pi)$. □

As a corollary, in the dishonest-majority setting $b \ge m/2$, $b$-private $(b, L)$-LD-PIR with the minimum list size implies single-server PIR with similar communication complexity. Indeed, as $b \ge m - b$, it implies an $(m - b)$-private $(m - b)$-server PIR protocol, from which we obtain a single-server PIR protocol by letting the server emulating the $m - b$ servers.

**Corollary 4.4.** *Assume that $m \le 2b$ and $|\mathcal{X}| \ge 6(L-1)$. If there exists a $b$-private $\ell$-round $m$-server $(b, L)$-LD-PIR protocol $\Pi$ with the minimum list size, then there exists a 1-private $\mathsf{negl}(\lambda)$-correct $\ell$-round 1-server PIR protocol $\Pi'$ such that $\mathrm{Comm}(\Pi') = \widetilde{O}(\mathrm{Comm}(\Pi))$.*

Damgård et al. [14] showed that the communication complexity of any information-theoretic $\mathsf{negl}(\lambda)$-correct single-server PIR protocol must be at least $N - \mathsf{negl}(\lambda)$, where $N$ is the database size. Therefore, Corollary 4.4 rules out the possibility of efficient $b$-private $(b, L)$-LD-PIR protocol with the minimum list size. Furthermore, even in the computational setting, it is supposedly impossible to construct an efficient $b$-private $(b, L)$-LD-PIR protocol from the minimal assumption of one-way functions. This is because the results of [16] showed that any $\mathsf{negl}(N)$-correct single-server PIR protocol $\Pi_1$ with $\mathrm{Comm}(\Pi_1) < N$ implies an honest-receiver oblivious transfer (OT) protocol with a security parameter $N$.

For example, in the dishonest-majority setting $b \ge m/2$, it rules out the possibility of efficient $b$-private $(b, 2)$-LD-PIR protocol. Similarly, there exists no efficient $b$-private $(b, 3)$-LD-PIR protocol if $2m/3 \le b <$

$3m/4$. In such cases, we must assume that the privacy threshold $t$ should be smaller than the number $b$ of errors to achieve sublinear communication complexity. Note that Corollary 4.4 does not rule out the possibility of efficient $b$-private $(b, L)$-LD-PIR protocols whose list size is allowed to be larger than the minimum. E.g., there remains the possibility of an efficient $b$-private $(b, 3)$-LD-PIR protocol for $m/2 \leq b < 2m/3$.

## 4.4 Towards the Difficulty of the Simulation-based Security

While our definition of LD-PIR is a game-based one, a natural question is whether LD-PIR can be defined in the simulation-based framework. A simulation-based definition is preferred in that it captures the security requirements of privacy and list decoding in a unified way. We consider the following ideal world on input a database $D$ and a client's index $x$ in the presence of a simulator $\mathcal{S}$ (i.e., an ideal-world adversary) corrupting a subset $T$ of servers:

1. A functionality $\mathcal{F}$ receives $D$ from servers and $x$ from the client.
2. $\mathcal{F}$ sends $D$ to $\mathcal{S}$ and receives some message $\mathsf{msg}$ from $\mathcal{S}$.
3. $\mathcal{F}$ returns to the client a list $\mathcal{L}$ such that
$$D_x \in \mathcal{L} \text{ and } |\mathcal{L}| \leq L. \tag{3}$$
4. Finally, $\mathcal{S}$ outputs a simulated view of the corrupted servers and the client outputs the list $\mathcal{L}$.

The security requirement is that the outputs of $\mathcal{S}$ and the client in the ideal world should be indistinguishable from those in the real world.

**Remark 4.5.** To discuss the difficulty of simulation-based security with generality, we do not specify $\mathcal{F}$ in full details in the above definition. The discussion below applies to any functionality $\mathcal{F}$ that returns a list $\mathcal{L}$ satisfying the condition (3). For example, a possible choice of $\mathcal{F}$ is that it receives an index $x$ from the client and a database $D^{(i)}$ from each server $\mathsf{S}_i$ for $i \in [m]$ (where $D^{(i)} = D$ if $\mathsf{S}_i$ is honest), and returns the $L$ most frequent elements in a multiset $\{D_x^{(1)}, \ldots, D_x^{(m)}\}$ to the client. This functionality corresponds to the case where $\mathsf{msg}$ consists of the $D^{(i)}$'s of corrupted servers.

We show that a protocol $\Pi$ satisfying the above simulation-based definition must be $b$-private and $(b, L)$-list-decodable in the sense of our game-based definition (i.e., Definition 4.1). First, it is easy to show that $\Pi$ is $(b, L)$-list-decodable. Namely, for $\Pi$ to successfully simulate the ideal world, the client's output list $\mathcal{L}$ in the real world should satisfy the condition (3) with overwhelming probability. Next, to show that $\Pi$ is $b$-private, let $x_0, x_1 \in [N]$, $D \in \mathcal{X}^N$, $T \subseteq [m]$ of size $b$, and $\mathcal{A}$ be an adversary who tries to guess the client's index. For $\alpha \in \{0, 1\}$, consider a real-world execution $\mathsf{REAL}_\alpha$ in which servers have $D$, the client has $x_\alpha$ and a real-world adversary computes responses of servers in $T$ in the same manner as $\mathcal{A}$ does. An ideal-world execution $\mathsf{IDEAL}_\alpha$ is the same as $\mathsf{REAL}_\alpha$ except that the real-world adversary is replaced with a simulator $\mathcal{S}$ who does not interact with the client. It is clear that the distribution of corrupted servers' views in $\mathsf{IDEAL}_0$ is identical with those in $\mathsf{IDEAL}_1$ since the simulated view is independent of $x_0$ or $x_1$. If $\Pi$ successfully simulates the ideal world, the distribution of corrupted servers' views in $\mathsf{REAL}_\alpha$ is indistinguishable from that in $\mathsf{IDEAL}_\alpha$. Therefore, the guess bit of $\mathcal{A}$ on input $x_\alpha$, which is a function of corrupted servers' views in $\mathsf{REAL}_\alpha$, is independent of $\alpha$. This implies the $b$-privacy of $\Pi$.

On the other hand, we have shown in Corollary 4.4 that a $b$-private $(b, L)$-LD-PIR protocol with the minimum list size implies a single-server PIR protocol with a similar communication complexity. Therefore, unless we assume single-server PIR or stronger cryptographic primitives, it is impossible to construct an efficient LD-PIR protocol with the minimum list size in the simulation-based framework. Note that there remains a possibility that we can define a meaningful simulation-based security for LD-PIR when the list size is allowed to be larger than the minimum.

Our game-based definition gets around the above difficulty by separating the number $b$ of errors from the privacy threshold $t$. That is, the size of each collusion of servers trying to learn a client's index $x$ should be at most $t$ while the total number $b$ of errors is allowed to be larger than $t$. We note that this corruption model follows those in the literature [6, 23, 26, 46].

# 5 Naive Construction Based on Polynomial Interpolation

In this section, we show a simple LD-PIR protocol with $o(N)$ communication for $m = b(L+1)/L + \Omega(Ldt)$. This protocol is derived by combining the regular PIR protocol in [42] and a list-decoding algorithm for multiplicity codes [28, 31, 32].

A starting point is the protocol of [42], whose formal descriptions are presented in Figure 13. For a parameter $d$, let $M$ be such that $\binom{M}{d} \geq N$, which can be chosen as $M = O(dN^{1/d})$. Then, we can find an injective function $E : [N] \to \{0,1\}^M$ such that the Hamming weight of $E(x) = (E(x)_1, \ldots, E(x)_M)$ is $d$. A client uses the function $E$ to encode his index $x \in [N]$ to a binary string $E(x) \in \{0,1\}^M$ of Hamming weight $d$. Each server encodes a database $D \in \mathcal{X}^N$ into an $M$-variate polynomial $F_D(z_1, \ldots, z_M)$ as

$$F_D(z_1, \ldots, z_M) = D_1 \boldsymbol{z}^{E(1)} + \cdots + D_N \boldsymbol{z}^{E(N)},$$

where $\boldsymbol{z}^{E(x)} = \prod_{j \in [M]: E(x)_j = 1} z_j$. Roughly speaking, the client distributes secret shares of $E(x)$ among servers who in response evaluate $F_D$ and its partial derivatives $(\partial F_D / \partial z_j)_{j \in [M]}$ on the shares. As a result, the client can compute $\boldsymbol{c} = (f(\alpha_i), f'(\alpha_i))_{i \in [m]}$ for a degree-$dt$ univariate polynomial $f$, where $f'$ is the derivative of $f$. If $dt \leq 2m - 1$, the client can recover $f$ from $\boldsymbol{c}$ based on polynomial interpolation and compute $f(0)$, which is equal to $F_D(E(x)) = D_x$.

To upgrade the protocol into LD-PIR, we observe that $\boldsymbol{c}$ forms a codeword in a *(univariate) multiplicity code*, which is a variant of Reed-Solomon codes augmented with derivatives. That is, if there are $b$ errors, $\boldsymbol{c}$ that the client computes has Hamming distance from $(f(\alpha_i), f'(\alpha_i))_{i \in [m]}$ less than or equal to $b$. We can then apply the existing list-decoding algorithms [28, 31, 32] and obtain a list of polynomials containing the correct $f$ from the erroneous codeword $\boldsymbol{c}$. Since [28, 31, 32] provided only asymptotic analyses allowing the list size $L$ to grow, we provide an analysis focusing on a concretely small list size $L$. We then have the following theorem. See Appendix H for more formal descriptions and proofs.

**Proposition 5.1.** *For a parameter $d \in \mathbb{N}$, suppose that*

$$m > \frac{L+1}{L} b + \frac{L+1}{4} dt \text{ and } m \geq b + \frac{L}{2} dt.$$

*Then, there exists a $t$-private 1-round $m$-server $(b, L)$-LD-PIR protocol $\Pi$ such that $\mathrm{Comm}(\Pi) = \widetilde{O}(N^{1/d} m d \log |\mathcal{X}|)$. The computation complexity of $\Pi$ is polynomial in $m$, $|\mathcal{X}|$ and $L$.*

However, the above protocol has the following drawbacks:

- The number of servers cannot achieve the list decoding bound $m > b(L+1)/L$ due to an additional factor of $\Omega(Ldt)$.
- The communication complexity is not comparable to that of the state-of-the-art regular PIR protocols with sub-polynomial communication complexity $N^{o(1)}$ [19, 20].

# 6 Generic Construction Based on an Exhaustive Search

In this section, we show a generic construction of LD-PIR protocols from regular PIR protocols. Our approach is to execute an error-detecting PIR protocol sufficiently many times in parallel. Consequently, the communication complexity of the resulting protocols cannot be bounded by a fixed polynomial in the number $m$ of servers.

First, for parameters $L$, $b$, and $k$, we define $m_L(b, k)$ as the smallest integer $m$ such that

$$m > \frac{L+1}{L} b \text{ and } m \geq b + k.$$

The following theorem shows a generic construction of LD-PIR protocols from error-detecting PIR protocols.

**Theorem 6.1.** *Let $\Pi$ be a $t$-private $(k-1)$-error-detecting 1-round $k$-server PIR protocol with failure probability $\epsilon$. Then, for any $m \geq m_L(b, k)$, there exists a $t$-private 1-round $m$-server $(b, L)$-LD-PIR protocol $\Pi'$ with failure probability $\binom{m}{k}\epsilon$ such that $\mathrm{Comm}(\Pi') = \mathrm{Comm}(\Pi) \cdot \binom{m}{k}$ and $\mathrm{Comp}(\Pi') = \mathrm{Comp}(\Pi) \cdot \widetilde{O}\left(\binom{m}{b}\binom{m-b}{k}\right)$.*

*Proof.* Let $\Pi'$ be an $m$-server PIR protocol obtained by running $\Pi$ between a client and each subset of $k$ servers in parallel. The communication cost of $\Pi'$ is $\mathrm{Comm}(\Pi') = \mathrm{Comm}(\Pi) \cdot \binom{m}{k}$. Clearly, $\Pi'$ is $t$-private since instances of $\Pi$ are independent.

We show that $\Pi'$ is $(b, L)$-list-decodable. Consider the following output algorithm $\mathsf{Output}'$ for $\Pi'$:

1. Take as input a tuple of $m$ answers, which is parsed as $\{(\mathsf{ans}_i^S)_{i\in S} : S \in \binom{[m]}{k}\}$, where $\mathsf{ans}_i^S$ is an answer returned by a server $\mathsf{S}_i$ in the instance of $\Pi$ executed by a client and a set $S$ of $k$ servers. In addition, take as input a state parsed as $\{\mathsf{st}^S : S \in \binom{[m]}{k}\}$, where $\mathsf{st}^S$ is a state returned by $\mathsf{Query}$ in the instance of $\Pi$ for $S$.
2. For each $S \in \binom{[m]}{k}$, let $y_S \leftarrow \mathsf{Output}(\mathsf{st}^S, (\mathsf{ans}_i^S)_{i\in S}) \in \mathcal{X} \cup \{\bot\}$.
3. Let $\mathcal{L}' = \emptyset$.
4. For each $H \in \binom{[m]}{m-b}$, if there exists $y \in \mathcal{X}$ such that $\{y_S : S \in \binom{H}{k}\} = \{y\}$, then set $z_H = y$ and add $(H, z_H)$ to $\mathcal{L}'$.
5. Find a maximal subset $\mathcal{L}''$ of $\mathcal{L}'$ such that for any $(H, z_H) \neq (H', z_{H'}) \in \mathcal{L}''$, it holds that $H \cap H' = \emptyset$. Such $\mathcal{L}''$ can be found efficiently by adding as many elements of $\mathcal{L}'$ as possible until no more elements can be added.
6. Output $\mathcal{L} = \{z_H : (H, z_H) \in \mathcal{L}''\}$.

Since $(L+1)(m-b) > m$, the set $[m]$ cannot be partitioned into more than $L$ subsets of size $m-b$. Thus, it holds that $|\mathcal{L}''| \leq L$ and $|\mathcal{L}| \leq L$.

Let $D \in \mathcal{X}^N$ be a database and $x \in [N]$ be a client's index. Let $H$ be a set of $m-b$ honest servers. Assume that the event that $D_x \notin \mathcal{L}$ happens. Then it must hold that $(H, z_H) \notin \mathcal{L}''$ since $z_H = D_x$. The maximality of $\mathcal{L}''$ implies that there exists $(H', z_{H'}) \in \mathcal{L}''$ such that $H \cap H' \neq \emptyset$. Since $z_{H'} \in \mathcal{L}$, it holds that $z_{H'} \in \mathcal{X} \setminus \{D_x\}$. Let $T$ be such that $T \in \binom{H'}{k}$ and $T \cap H \neq \emptyset$. There actually exists such $T$ since $H \cap H' \neq \emptyset$. Since $\mathsf{ans}_i^T$ is a correct answer for $i \in T \cap H$ and $\Pi$ is $(k-1)$-error-detecting, it holds except with negligible probability $\epsilon \in \mathsf{negl}(\lambda)$ that $y_T = \mathsf{Output}(\mathsf{st}^T, (\mathsf{ans}_i^T)_{i\in T}) \in \{D_x, \bot\}$. This contradicts with $z_{H'} \in \mathcal{X} \setminus \{D_x\}$ since $y_T \in \{y_S : S \in \binom{H'}{k}\} = \{z_{H'}\}$. We conclude that the failure probability of $\Pi'$ is upper bounded by the probability that $y_T \notin \{D_x, \bot\}$ for some $T \in \binom{[m]}{k}$, which is at most $\binom{m}{k}\epsilon$. □

Applying the compiler of [23], we obtain a generic construction of LD-PIR protocols from regular PIR protocols.

**Corollary 6.2.** *Let $\Pi$ be a $t$-private 1-round $k$-server PIR protocol. Then, for any $m \geq m_L(b, k)$, there exists a $t$-private 1-round $m$-server $(b, L)$-LD-PIR protocol $\Pi'$ such that* $\mathrm{Comm}(\Pi') = \mathrm{Comm}(\Pi) \cdot \widetilde{O}\left(\binom{m}{k}k^3\right)$ *and* $\mathrm{Comp}(\Pi') = \mathrm{Comp}(\Pi) \cdot \widetilde{O}\left(\binom{m}{b}\binom{m-b}{k}k^3\right)$.

*Proof.* $\Pi$ can be transformed into a $t$-private $(k-1)$-error-detecting 1-round $k$-server PIR protocol $\Pi_{\mathrm{ED}}$ such that $\mathrm{Comm}(\Pi_{\mathrm{ED}}) = \widetilde{O}(\mathrm{Comm}(\Pi) \cdot k^2)$. More precisely, according to [23], $\mathrm{Comm}(\Pi_{\mathrm{ED}}) = \mathrm{Comm}(\Pi) \cdot O(\kappa)$ and the failure probability of $\Pi_{\mathrm{ED}}$ is at most $\exp(-\kappa/k^2)$ for a parameter $\kappa \in \mathbb{N}$. From Theorem 6.1, $\Pi_{\mathrm{ED}}$ can be transformed into a $t$-private 1-round $m$-server $(b, L)$-LD-PIR protocol $\Pi'$ such that $\mathrm{Comm}(\Pi') = \mathrm{Comm}(\Pi_{\mathrm{ED}}) \cdot \binom{m}{k} = \mathrm{Comm}(\Pi) \cdot O(\binom{m}{k}\kappa)$ and the failure probability is $\binom{m}{k} \cdot \exp(-\kappa/k^2) = \exp(-\kappa/k^2 + k \ln m)$. Set $\kappa = \Theta(k^2((\ln \lambda)^2 + k \ln m)) = \widetilde{\Theta}(k^3(\ln \lambda)^2)$. Then, $\mathrm{Comm}(\Pi') = \mathrm{Comm}(\Pi) \cdot \widetilde{O}(\binom{m}{k}k^3)$ and the failure probability is negligible in $\lambda$. The computation complexity of $\Pi'$ is $O(\binom{m}{k}\binom{m-b}{k}\kappa) = \widetilde{O}(\binom{m}{k}\binom{m-b}{k}k^3)$ times larger than that of $\Pi$. □

By instantiating the above construction with the PIR protocols in [19, 20], we obtain $(b, L)$-LD-PIR protocols with sub-polynomial communication complexity $N^{o(1)}$. Furthermore, the number $m_L(b, k)$ of servers reaches the list decoding bound $m > b(L+1)/L$ for a sufficiently larger $b$, which is the minimum number of servers required for $(b, L)$-LD-PIR. We explain more details on concrete instantiations in Section 8.

**Remark 6.3.** The construction in Corollary 6.2 requires an additional bound $m \geq b + k$ on the number $m$ of servers in addition to the list decoding bound but this bound seems the best possible. Assume for contradiction that there exists a generic compiler that transforms any $k$-server PIR protocol to an $m$-server $(b, L)$-LD-PIR protocol with $m = m_L(b, k-1)$. Setting $b := (L-1)(k-1)$, the compiler yields an $m$-server $(b, L)$-LD-PIR protocol with $m = L(k-1)$. Note that since $b(L+1)/L < m \leq bL/(L-1)$, the resulting LD-PIR protocol has the minimum list size. We can then apply the compiler in Theorem 4.3, obtaining a PIR protocol whose number of servers is $m - b = k - 1$. In summary, a $(k-1)$-server PIR protocol would be obtained from any $k$-server PIR protocol via the compilers of Corollary 6.2 and Theorem 4.3. We conclude that the bound $m \geq b + k$ is unavoidable unless there is a generic method to reduce the number of servers in regular PIR protocols. Such a method has not been found so far.

# 7 Generic Constructions with $\mathsf{poly}(m)$ Communication Overheads

Since the construction in Corollary 6.2 incurs large communication and computation overheads in $m$, it can only deal with a small number of servers. In this section, we propose different generic constructions whose communication overheads are bounded by a fixed polynomial in $m$.

## 7.1 Two-round Construction

Let $\Pi$ be a $(t+1)$-private $(m-b)$-robust $m$-server PIR protocol. First, we construct a 2-round sub-protocol $\Pi_0$ based on $\Pi$ as follows. In the first round, a client $\mathsf{C}$ runs $\Pi$ with $m$ servers $\mathsf{S}_1, \ldots, \mathsf{S}_m$ as usual. Let $a_i$ be the answer returned by $\mathsf{S}_i$. In the second round, $\mathsf{C}$ chooses one server $\mathsf{S}_\ell$ uniformly at random and sends all servers but $\mathsf{S}_\ell$ the query $\mathsf{que}_\ell$ which he sent to $\mathsf{S}_\ell$ in the first round. Each server replies to $\mathsf{que}_\ell$ as the server $\mathsf{S}_\ell$ would reply to $\mathsf{que}_\ell$ in the first round. This is possible because the answer algorithm of $\Pi$ is deterministic. For $i \in [m] \setminus \{\ell\}$, let $b_i$ be the answer which $\mathsf{S}_i$ returns as an answer to $\mathsf{que}_\ell$. We set $b_\ell = a_\ell$. In the sub-protocol $\Pi_0$, the output of $\mathsf{C}$ is $(a_i, b_i)_{i \in [m]}$. It is important to note that $b_i = b_j = \mathsf{Answer}(D, \mathsf{que}_\ell)$ for all $i, j \in H$, where $H$ is the set of honest servers, since all honest servers return the same reply to $\mathsf{que}_\ell$. Furthermore, if the server $\mathsf{S}_\ell$ returns an incorrect answer $a_\ell$ in the first round, the answer conflicts with the answer computed by any honest server, i.e., $b_\ell = a_\ell \neq b_i$ for all $i \in H$.

Our LD-PIR protocol $\Pi'$ is obtained by running the sub-protocol $\Pi_0$ sufficiently many (say, $\kappa$) times in parallel. For $u \in [\kappa]$, let $\ell^{(u)}$ be an index chosen by $\mathsf{C}$ in the $u$-th instance of $\Pi_0$. For $i \in [m]$, define $v_i = (b_i^{(u)})_{u \in [\kappa]}$ be a tuple of second-round answers of $\mathsf{S}_i$. To obtain a list $\mathcal{L}$, $\mathsf{C}$ partitions the set of $m$ servers into equivalence classes under the equivalence relation: $\mathsf{S}_i \sim \mathsf{S}_j \stackrel{\text{def}}{\iff} v_i = v_j$. Let $C_1, \ldots, C_h$ be the equivalence classes of size at least $m-b$. Note that there is an equivalence class that includes $H$ since $v_i = v_j$ for all $i, j \in H$ and $|H| = m - b$ For each equivalence class $C_g$, $\mathsf{C}$ computes $y_g^{(1)}, \ldots, y_g^{(\kappa)} \in \mathcal{X}$ each of which is computed from the first-round answers $(a_i^{(u)})_{i \in C_g}$ of servers in $C_g$. Then, $\mathsf{C}$ computes the majority $y_g$ of the $\kappa$ values $y_g^{(1)}, \ldots, y_g^{(\kappa)}$. Finally, $\mathsf{C}$ outputs a list of candidate values $\mathcal{L} = \{y_g : g \in [h]\}$.

**Sketch of Proof.** Regarding privacy, during the execution of $\Pi_0$, an adversary learns $t$ queries that corrupted servers receive in the first round plus one query $\mathsf{que}_\ell$ that all servers receive in the second round. Since $\Pi$ is $(t+1)$-private, the adversary thus learns no information on the client's index. Since $\Pi$ executes $\Pi_0$ $\kappa$ times independently, the $t$-privacy of $\Pi'$ follows.

We show that $\Pi'$ is $(b, L)$-list-decodable. First, if $m > b(L+1)/L$, then we can see that $m < (L+1)(m-b)$. On the other hand, it holds that $m \geq |C_1| + \cdots + |C_h| \geq h(m-b)$. Hence, the number of equivalence classes of size at least $m - b$ is at most $L$ and we have that $|\mathcal{L}| \leq L$. Next, to see that $D_x \in \mathcal{L}$, assume otherwise that $y_g \neq D_x$ for all $g \in [h]$. Without loss of generality, we may assume that $H \subseteq C_1$. If $y_1 \neq D_x$, it follows by definition of $y_1$ that at least $\kappa/2$ values of $y_1^{(1)}, \ldots, y_1^{(\kappa)}$, say $y_1^{(1)}, \ldots, y_1^{(\kappa/2)}$, are not equal to $D_x$. Since $y_1^{(u)}$ is computed from $(a_i^{(u)})_{i \in C_1}$ for every $u \in [\kappa/2]$, there exists at least one incorrect answer in $(a_i^{(u)})_{i \in C_1 \setminus H}$. Hence, the $|C_1 \setminus H| \cdot \kappa/2$ answers $(a_i^{(1)})_{i \in C_1 \setminus H}, \ldots, (a_i^{(\kappa/2)})_{i \in C_1 \setminus H}$ contain at least $\kappa/2$ incorrect answers. Therefore, since $|C_1 \setminus H| \leq b$, there exists a corrupted server $\mathsf{S}_i \in C_1 \setminus H$ such that $(a_i^{(1)}, \ldots, a_i^{(\kappa/2)})$ contains at least $\kappa/(2b)$ incorrect answers. Let $K \subseteq [\kappa/2]$ be the set of $\kappa/(2b)$ instances of $\Pi_0$ where $\mathsf{S}_i$ returns incorrect answers, i.e., $K = \{u \in [\kappa/2] : a_i^{(u)} \text{ is incorrect}\}$. On the other hand, for the server $\mathsf{S}_i$ to be

included in the same equivalence class $C_1$ as honest servers, $\mathsf{S}_i$ must not be chosen as a target server in the $u$-th instance of $\Pi_0$ for any $u \in K$. This is because otherwise, the answer $a_i^{(u)}$ is compared with the answers $b_j^{(u)}$ of honest servers $\mathsf{S}_j \in H$ and then $\mathsf{S}_i$ is not included in $C_1$. The probability of this event is at most $(1-1/m)^{\kappa/(2b)} \leq \exp(-\kappa/(2bm))$, which can be made negligible by choosing $\kappa = \widetilde{\Theta}(mb)$.

Now, we have the following theorem. The formal proof is given in Appendix C.

**Theorem 7.1.** *Let $\Pi$ be a $k$-robust $(t+1)$-private 1-round $m$-server PIR protocol. Then, for any $m \geq m_L(b,k)$, there exists a 2-round $t$-private $m$-server $(b,L)$-LD-PIR protocol $\Pi'$ such that* $\mathrm{Comm}(\Pi') = \mathrm{Comm}(\Pi) \cdot \widetilde{O}(mb)$ *and* $\mathrm{Comp}(\Pi') = \mathrm{Comp}(\Pi) \cdot \widetilde{O}(m^2)$.

Applying the compilers of [6], we obtain generic constructions of LD-PIR protocols from regular PIR protocols.

**Corollary 7.2.** *Let $\Pi$ be a $(t+1)$-private 1-round $k$-server PIR protocol. Then, for any $m \geq m_L(b,k)$, there exists a 2-round $t$-private $m$-server $(b,L)$-LD-PIR protocol $\Pi'$ such that* $\mathrm{Comm}(\Pi') = \mathrm{Comm}(\Pi) \cdot \widetilde{O}(ke^k m^2 b)$ *and* $\mathrm{Comp}(\Pi') = \mathrm{Comp}(\Pi) \cdot \widetilde{O}(ke^k m^3)$

**Corollary 7.3.** *Let $\Pi$ be a $(t+1)$-private 1-round $k$-server PIR protocol. Let $k'$ be such that $k'/\ln k' \geq k$. Then, for any $m \geq m_L(b,k')$, there exists a 2-round $t$-private $m$-server $(b,L)$-LD-PIR protocol $\Pi'$ such that* $\mathrm{Comm}(\Pi') = \mathrm{Comm}(\Pi) \cdot \widetilde{O}(km^2 b)$ *and* $\mathrm{Comp}(\Pi') = \mathrm{Comp}(\Pi) \cdot \widetilde{O}(km^3)$.

## 7.2 One-round Construction in the Local Error Model

We show a one-round variant of the previous two-round construction, assuming the local error model defined below. In this model, a corrupted server injects errors based on the views of a *subset* of the corrupted servers, rather than on the joint view of all $b$ corrupted servers. This restricted model remains meaningful in scenarios where corrupted servers are non-colluding and each of them modifies its answer solely based on its own view. This setting naturally generalizes the model that was introduced by [21] originally in the context of error-detecting PIR.

**Definition 7.4.** *Let $\Pi = (\mathsf{Query}, \mathsf{Answer}, \mathsf{Output})$ be an $\ell$-round $m$-server PIR protocol. Let $v \in \mathbb{N}$ be a parameter such that $v \leq b$. We say that $\Pi$ is* $(b,L)$-list-decodable in the $v$-local error model *if* $\mathsf{Output}$ *is allowed to output a subset $\mathcal{L} \subseteq \mathcal{X}$ and $\Pi$ satisfies the following requirements: There exists a negligible function $\epsilon = \epsilon(\lambda)$ such that for any $\lambda \in \mathbb{N}$ and any stateful algorithm $\mathcal{A}$, it holds that $\Pr[\mathsf{LD}_{\Pi,\mathcal{A}}(\lambda) = 1] \leq \epsilon$, where $\mathsf{LD}_{\Pi,\mathcal{A}}(\lambda)$ is the output of the following experiment $\mathrm{EXPT}^{\mathrm{LD}}_{\Pi,\mathcal{A}}(\lambda)$:*

1. *$\mathcal{A}(1^\lambda)$ outputs $(x, D, (T_k, \mathcal{A}_k)_{k\in[h]})$, where $D \in \mathcal{X}^N$, $x \in [N]$, $(T_k)_{k\in[h]}$ are pairwise disjoint subsets of $[m]$ such that $|T_k| \leq v$ for any $k \in [h]$ and $|T_1 \cup \cdots \cup T_h| \leq b$, and $(\mathcal{A}_k)_{k\in[h]}$ are independent stateful algorithms.*
2. *Let $((\mathsf{que}_i)_{i\in[m]}, \mathsf{st}) \leftarrow \mathsf{Query}(1^\lambda, x)$. For each $k \in [h]$, give $(\mathsf{que}_i)_{i\in T_k}$ to $\mathcal{A}_k$.*
3. *Each $\mathcal{A}_k$ outputs $(\mathsf{ans}_i)_{i\in T_k}$. Let $\mathsf{ans}_i \leftarrow \mathsf{Answer}(D, \mathsf{que}_i)$ for $i \notin T_1 \cup \cdots \cup T_h$.*
4. *Let $\mathcal{L} \leftarrow \mathsf{Output}(\mathsf{st}, (\mathsf{ans}_i)_{i\in[m]})$. Return 0 if $D_x \in \mathcal{L}$ and $|\mathcal{L}| \leq L$, and otherwise return 1.*

To obtain one-round LD-PIR, we reduce the number of rounds in the previous construction by letting the client $\mathsf{C}$ sends servers first-round and second-round queries simultaneously. Specifically, $\mathsf{C}$ computes two independent queries $(q_i)_{i\in[m]} \leftarrow \mathsf{Query}(x)$ and $(r_i)_{i\in[m]} \leftarrow \mathsf{Query}(x)$. $\mathsf{C}$ chooses one server $\mathsf{S}_\ell$ uniformly at random and sends each server $\mathsf{S}_i$ two queries $(q_i, r_\ell)$. We call each $q_i$ a *computation query* as it will be used to compute outputs and $r_\ell$ a *verification query* as it will be used to verify the consistency of answers. Each server $\mathsf{S}_i$ replies to $q_i$ as usual and replies to $r_\ell$ as $\mathsf{S}_\ell$ would reply to $r_\ell$. Let $a_i, b_i$ be answers returned by $\mathsf{S}_i$ to $q_i, r_\ell$, respectively. Note that if all answers are correct, then $b_1 = \cdots = b_m = \mathsf{Answer}(D, r_\ell)$.

A list-decoding procedure proceeds similarly to the previous construction. Specifically, $\mathsf{C}$ executes the above sub-protocol in parallel $\kappa$ times. Then, based on the servers' responses $b_i$ to the verification queries $r_\ell$, $\mathsf{C}$ partitions the servers into equivalence classes $C_1, \ldots, C_h$. Finally, for each class $C_g$, $\mathsf{C}$ computes an output $y_g$ from the answers $(a_i)_{i\in C_g}$, and returns the collection $\{y_g : g \in [h]\}$ of such outputs as the final list.

An important distinction is that the target server $\mathsf{S}_\ell$ must not be able to determine whether a given query will be used for output computation or for answer verification. Otherwise, $\mathsf{S}_\ell$ could cause errors by responding honestly to verification queries while providing incorrect answers to computation queries. To prevent this attack, we randomize the order of verification and computation queries. As a result, to cause $\mathsf{C}$ to output an incorrect list, a corrupted server must inject errors in at least $\kappa/(2b)$ instances of the sub-protocol from the same argument in the previous two-round construction. Furthermore, the server can succeed only if, in each of those instances, it is either not selected as the target server or correctly guesses the computation queries. The probability of successfully doing so is upper bounded by approximately $(1-1/(2m))^{\kappa/(2b)} = \exp(-\kappa/(4mb))$. Therefore, it suffices to set $\kappa = \widetilde{\Theta}(mb)$.

We note that the resulting protocol is $(b, L)$-LD-PIR in the 1-local error model since if a target server sees queries for other corrupted servers, then it can trivially distinguish between verification and computation queries. This generic construction can be built upon $t$-private PIR protocols instead of $(t+1)$-private ones since the verification queries $r_\ell$ are generated independently of the computation queries $q_i$.

In summary, we have the following theorem. We defer the formal descriptions and proofs to Appendix D.

**Theorem 7.5.** *Let $\Pi$ be a $k$-robust $t$-private 1-round $m$-server PIR protocol. Then, for any $m \geq m_L(b,k)$, there exists a $t$-private 1-round $m$-server $(b, L)$-LD-PIR protocol $\Pi'$ in the 1-local error model such that $\mathrm{Comm}(\Pi') = \mathrm{Comm}(\Pi) \cdot \widetilde{O}(mb)$ and $\mathrm{Comp}(\Pi') = \mathrm{Comp}(\Pi) \cdot \widetilde{O}(m^2)$*

Applying the compilers of [6], we obtain the following corollaries.

**Corollary 7.6.** *Let $\Pi$ be a $t$-private 1-round $k$-server PIR protocol. Then, for any $m \geq m_L(b,k)$, there exists a $t$-private 1-round $m$-server $(b, L)$-LD-PIR protocol $\Pi'$ in the 1-local error model such that $\mathrm{Comm}(\Pi') = \mathrm{Comm}(\Pi) \cdot \widetilde{O}(ke^k m^2 b)$ and $\mathrm{Comp}(\Pi') = \mathrm{Comp}(\Pi) \cdot \widetilde{O}(ke^k m^3)$.*

**Corollary 7.7.** *Let $\Pi$ be a $t$-private 1-round $k$-server PIR protocol. Let $k'$ be such that $k'/\ln k' \geq k$. Then, for any $m \geq m_L(b,k')$, there exists a $t$-private 1-round $m$-server $(b, L)$-LD-PIR protocol $\Pi'$ in the 1-local error model such that $\mathrm{Comm}(\Pi') = \mathrm{Comm}(\Pi) \cdot \widetilde{O}(km^2 b)$ and $\mathrm{Comp}(\Pi') = \mathrm{Comp}(\Pi) \cdot \widetilde{O}(km^3)$.*

## 7.3 One-round Construction in the Non-local Error Model

We present another one-round variant in the non-local error model (i.e., satisfying Definition 4.1) at the cost of increasing the client-side computation complexity. A high-level idea is that a client $\mathsf{C}$ sends a verification query $r_\ell$ only to another randomly chosen server $\mathsf{S}_j$, instead of sending $r_\ell$ to all servers. Specifically, after generating computation queries $(q_i)_{i\in[m]}$ and verification queries $(r_i)_{i\in[m]}$, $\mathsf{C}$ chooses a pair $(\mathsf{S}_\ell, \mathsf{S}_j)$ of servers uniformly at random. $\mathsf{C}$ then sends $(q_j, r_\ell)$ to $\mathsf{S}_j$ and sends $(q_i, r_i)$ to other servers $\mathsf{S}_i$ in a random order. That is, each server $\mathsf{S}_i$ receives (a row-permuted version of) the $i$-th column of the following matrix

$$\begin{pmatrix} q_1, & \ldots, & q_{j-1}, & q_j, & q_{j+1}, & \ldots, & q_m \\ r_1, & \ldots, & r_{j-1}, & \boxed{r_\ell}, & r_{j+1}, & \ldots, & r_m \end{pmatrix}. \tag{4}$$

$\mathsf{C}$ executes this sub-protocol in parallel $\kappa$ times.

Accordingly, we modify the list decoding algorithm as follows. After receiving answers, $\mathsf{C}$ first constructs a graph $G$ on $m$ nodes each representing a server. Initially, $\mathsf{C}$ sets $G$ to a complete graph. For each $u \in [\kappa]$, let $(\ell^{(u)}, j^{(u)})$ be a pair of servers chosen in the $u$-th instance. $\mathsf{C}$ checks if their answers to the verification query $r_{j^{(u)}}$ are equal. If their answers are not equal, then $\mathsf{C}$ removes an edge $(\ell^{(u)}, j^{(u)})$ from $G$. $\mathsf{C}$ iterates this process for all $u \in [\kappa]$.

Note that if both servers $(\ell^{(u)}, j^{(u)})$ behave honestly, then the verification always passes and the edge between them remains. In particular, the set $H$ of honest servers forms a *clique* of size $m-b$ in the graph $G$. Recall that a clique is a subset of vertices such that every pair of vertices in it are adjacent. Based on this observation, $\mathsf{C}$ finds any clique $C_1$ of size $m-b$ in $G$ and removes the nodes in $C_1$ from $G$. Then, $\mathsf{C}$ finds any clique $C_2$ of size $m-b$ in the resulting graph $G'$ and continues this procedure. Finally, $\mathsf{C}$ obtains at most $L$ cliques. For each clique, he computes an output from the corresponding answers to the computation queries.

We analyze the failure probability. Similarly to the previous constructions, the protocol fails only if at least one corrupted server $\mathsf{S}_\ell$ injects errors in at least $\kappa/(2b)$ instances. Furthermore, $\mathsf{S}_\ell$ must be included in the clique including $H$ and hence there must remain an edge between $\mathsf{S}_\ell$ and an honest server $\mathsf{S}_j \in H$. This means that in each of the $\kappa/(2b)$ instances, one of the following events must occur:

- The pair $(\mathsf{S}_\ell, \mathsf{S}_j)$ is not selected by $\mathsf{C}$.
- $\mathsf{S}_\ell$ returns a correct answer to the verification query.

The probability of the first event is $1 - 1/(m(m-1))$. It follows from the query structure in (4) that $\mathsf{S}_\ell$ cannot distinguish between computation and verification queries even if it colludes with other corrupted servers. The probability of the second event is thus $1/2$. Therefore, the failure probability is upper bounded by by approximately $(1 - 1/(2m(m-1)))^{\kappa/(2b)} \leq \exp(-\kappa/(4m^2b))$, which is negligible if we set $\kappa = \widetilde{\Theta}(m^2b)$.

We note that $\mathsf{C}$ needs to find a clique of size $m - b$ in a graph with $m$ nodes. If the brute-force search algorithm is used, the computation complexity of list decoding is $O(\binom{m}{b}m^2)$. Since the problem of finding cliques is conjectured to have no fixed polynomial-time algorithm [18], it is supposedly hard to find a list-decoding algorithm running in fixed polynomial time $\mathsf{poly}(m)$ in the above approach.

In summary, we have the following theorem. We defer the formal descriptions and proofs to Appendix D.

**Theorem 7.8.** *Let $\Pi$ be a $k$-robust $t$-private 1-round $m$-server PIR protocol. Then, for any $m \geq m_L(b,k)$, there exists a $t$-private 1-round $m$-server $(b,L)$-LD-PIR protocol $\Pi'$ such that $\mathrm{Comm}(\Pi') = \mathrm{Comm}(\Pi) \cdot \widetilde{O}(m^2b)$ and $\mathrm{Comp}(\Pi') = \mathrm{Comp}(\Pi) \cdot \widetilde{O}(m^2b) + L\tau_{m,m-b}$, where $\tau_{m,m-b}$ is the computation complexity of finding a clique of size $m-b$ in a graph with $m$ nodes.*

Applying the compilers of [6], we obtain generic constructions of LD-PIR protocols from regular PIR protocols. Note that $\tau_{m,k} = O(\binom{m}{k}k^2)$ if the brute-force algorithm is used.

**Corollary 7.9.** *Let $\Pi$ be a $t$-private 1-round $k$-server PIR protocol. Then, for any $m \geq m_L(b,k)$, there exists a $t$-private 1-round $m$-server $(b,L)$-LD-PIR protocol $\Pi'$ such that $\mathrm{Comm}(\Pi') = \mathrm{Comm}(\Pi) \cdot \widetilde{O}(ke^km^3b)$ and $\mathrm{Comp}(\Pi') = \mathrm{Comp}(\Pi) \cdot \widetilde{O}(ke^km^3b) + O\left(\binom{m}{b}m^2L\right)$.*

**Corollary 7.10.** *Let $\Pi$ be a $t$-private 1-round $k$-server PIR protocol. Let $k'$ be such that $k'/\ln k' \geq k$. Then, for any $m \geq m_L(b,k')$, there exists a $t$-private 1-round $m$-server $(b,L)$-LD-PIR protocol $\Pi'$ such that $\mathrm{Comm}(\Pi') = \mathrm{Comm}(\Pi) \cdot \widetilde{O}(km^3b)$ and $\mathrm{Comp}(\Pi') = \mathrm{Comp}(\Pi) \cdot \widetilde{O}(km^3b) + O\left(\binom{m}{b}m^2L\right)$.*

# 8 Instantiations

We instantiate the generic constructions from Sections 6 and 7 with the currently best-known PIR protocols. To simplify discussion, we focus on one-round LD-PIR protocols. Furthermore, in this section, we focus on instantiations from Dvir-Gopi's PIR [19] and defer instantiations from other protocols [20, 42] to Appendix F.

The protocol of Dvir and Gopi [19] achieves sub-polynomial communication complexity in the database size $N$. The original protocol in [19] assumes $t = 1$ but can be extended to a $t$-private protocol for $t \geq 2$ by using the privacy amplification technique in [3].

**Proposition 8.1.** *There exists a 1-round $t$-private $2^t$-server PIR protocol $\mathsf{DG}_t$ such that $\mathrm{Comm}(\mathsf{DG}_t) = O(h_N^t 2^t \log|\mathcal{X}|)$, where $h_N = \exp(O(\sqrt{\log N \log\log N})) = N^{o(1)}$ is the value specified by an underlying matching vector code (see [19] for details).*

We introduce notations describing our generic constructions of LD-PIR:

- $\mathsf{Exh}$ denotes the construction from Corollary 6.2.
- $\mathsf{Local1}$ denotes the construction from Corollary 7.6.
- $\mathsf{Local2}$ denotes the construction from Corollary 7.7.
- $\mathsf{Clique1}$ denotes the construction from Corollary 7.9.
- $\mathsf{Clique2}$ denotes the construction from Corollary 7.10.

We denote by $\mathsf{X}\text{-}\mathsf{DG}_t$ an LD-PIR protocol obtained by applying a generic construction

$$\mathsf{X} \in \{\mathsf{Exh}, \mathsf{Local1}, \mathsf{Local2}, \mathsf{Clique1}, \mathsf{Clique2}\}$$

Table 2: One-round LD-PIR protocols obtained from Dvir-Gopi's PIR

| | | Overhead | | |
|---|---|---|---|---|
| Protocol | Servers | Communication | Computation | Remark |
| $\mathsf{Exh}\text{-}\mathsf{DG}_t$ | $m_L(b, 2^t)$ | $\binom{m}{2^t}2^{3t}$ | $\binom{m}{b}\binom{m-b}{2^t}2^{3t}$ | |
| $\mathsf{Local1}\text{-}\mathsf{DG}_t$ | $m_L(b, 2^t)$ | $m^2b2^te^{2^t}$ | $m^2b2^te^{2^t}$ | 1-local error model |
| $\mathsf{Local2}\text{-}\mathsf{DG}_t$ | $m_L(b, t2^{t+1})$ | $m^2bt2^t$ | $m^2bt2^t$ | 1-local error model |
| $\mathsf{Clique1}\text{-}\mathsf{DG}_t$ | $m_L(b, 2^t)$ | $m^3b2^te^{2^t}$ | $\binom{m}{b}m^3b2^te^{2^t}L$ | |
| $\mathsf{Clique2}\text{-}\mathsf{DG}_t$ | $m_L(b, t2^{t+1})$ | $m^3bt2^t$ | $\binom{m}{b}m^3bt2^tL$ | |

$m$ is the number of servers, $b$ is the number of errors, $L$ is the list size, and $t$ is a privacy threshold. "Servers" refers the minimum number of servers required and $m_L(b, k)$ denotes the smallest integer $n$ such that $n > b(L+1)/L$ and $n \geq b + k$. "Overhead" refers to the ratio of the communication (resp. computation) complexity relative to that of $\mathsf{DG}_t$, omitting polylogarithmic factors in $m$ and $\lambda$.

to $\mathsf{DG}_t$. We use the fact that $k'/\ln k' \geq 2^t$ if $k' = t2^{t+1}$ to apply $\mathsf{X} \in \{\mathsf{Local2}, \mathsf{Clique2}\}$. We summarize the performance of these protocols in Table 2.

**Number of Servers.** Among all of them, $\mathsf{Exh}\text{-}\mathsf{DG}_t$, $\mathsf{Local1}\text{-}\mathsf{DG}_t$, and $\mathsf{Clique1}\text{-}\mathsf{DG}_t$ achieve the minimum number of servers $m \geq m_L(b, 2^t)$. Notably, they even achieve the minimum number of servers required for $(b, L)$-list-decoding, i.e., $m > b(L+1)/L$, for a sufficiently large number of errors $b \geq L2^t$. The case of $(b, m) = (2, 4)$ constitutes the smallest example in which LD-PIR can tolerate a majority of corrupted servers while achieving non-trivial communication complexity of $N^{o(1)}$. More concretely, Figure 1 shows the changes in the minimum number $m$ of servers required by each LD-PIR protocol with respect to the number $b$ of errors. $\mathsf{Baseline}_{t,d}$ denotes the naive protocol in Section 5. We fix $t = 1$, $d = 3$, and set $L \in \{2, 3\}$. Among all of them, the number of servers in $\mathsf{X}\text{-}\mathsf{DG}_t$ ($\mathsf{X} \in \{\mathsf{Exh}, \mathsf{Local1}, \mathsf{Clique1}\}$) becomes smaller than the error-correcting bound $2b+1$ and approaches the list decoding bound $b(L+1)/L = 2b/3$ more quickly than others. Then, the number of servers in $\mathsf{X}\text{-}\mathsf{DG}_t$ ($\mathsf{X} \in \{\mathsf{Local2}, \mathsf{Clique2}\}$) and $\mathsf{Baseline}_{t,d}$ falls below the error-correcting bound in this order. The number of servers in $\mathsf{Baseline}_{t,d}$ never converges to the list decoding bound.

**Communication Overhead.** $\mathsf{Clique2}\text{-}\mathsf{DG}_t$ achieves the asymptotically lowest communication and computation complexity in the non-local error model and $\mathsf{Local2}\text{-}\mathsf{DG}_t$ is the lowest in the local error model. More concretely, Figure 2 shows the ratio of the communication complexity of each LD-PIR protocol $\mathsf{X}\text{-}\mathsf{DG}_t$ ($\mathsf{X} \in \{\mathsf{Exh}, \mathsf{Local1}, \mathsf{Local2}, \mathsf{Clique1}, \mathsf{Clique2}\}$), relative to that of the underlying regular PIR protocol $\mathsf{DG}_t$. We fix $L = 2$ and the failure probability to $\epsilon = 2^{-60}$. For $t = 1$, the factor of $\binom{m}{2^t} = O(m^2)$ remains small and $\mathsf{Exh}\text{-}\mathsf{DG}_t$ has the lowest communication complexity. On the other hand, as $t$ and $b$ grow, the factor of $\binom{m}{2^t} = O(m^{2^t})$ becomes prohibitive and $\mathsf{X}\text{-}\mathsf{DG}_t$ for $\mathsf{X} \in \{\mathsf{Local1}, \mathsf{Local2}, \mathsf{Clique1}, \mathsf{Clique2}\}$ achieves much lower communication complexity. $\mathsf{Local2}\text{-}\mathsf{DG}_t$ and $\mathsf{Clique2}\text{-}\mathsf{DG}_t$ tend to exhibit lower communication complexity than $\mathsf{Local1}\text{-}\mathsf{DG}_t$ and $\mathsf{Clique1}\text{-}\mathsf{DG}_t$, as they eliminate the doubly exponential factor $e^{2^t}$.

The advantages of our constructions are summarized as follows:

- Exh, Local1, and Clique1 achieve the smallest number of servers.
- When the number $b$ of errors remains small, Exh achieves the lowest communication complexity.
- When $b$ grows, Local2 achieves the lowest communication complexity.

# 9 Unique Decoding with a Trusted Setup

It is known that unique decoding is impossible in the dishonest-majority setting, i.e., $m \leq 2b$. We show that if a trusted party is assumed who authenticates a database in the setup, then LD-PIR can be upgraded into uniquely decodable PIR even when $m \leq 2b$. We notice that protocols in this section guarantee unique decoding for computationally bounded adversaries.

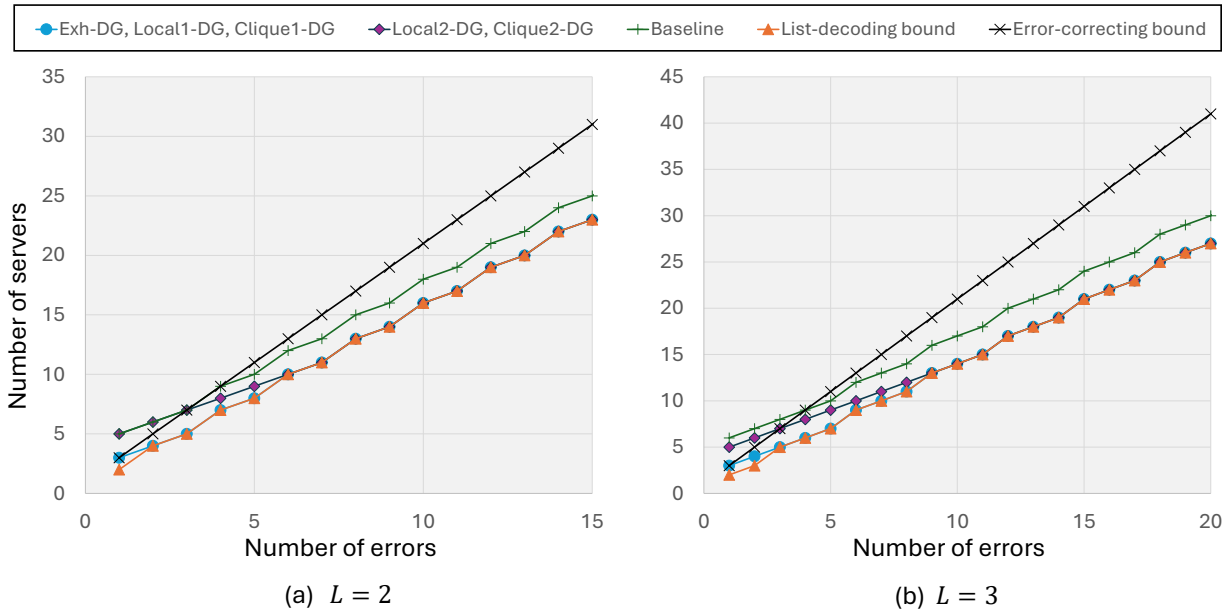


Figure 1: The number of servers in one-round LD-PIR in the setting of $L \in \{2, 3\}$ and $t = 1$ (an excerpt from Figure 10)

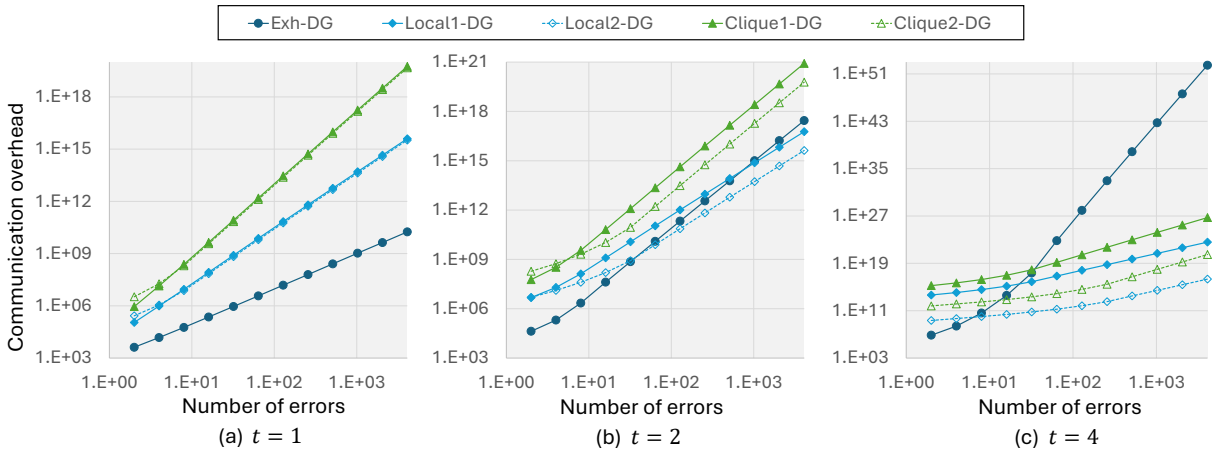


Figure 2: Communication overhead of one-round LD-PIR obtained from Dvir-Gopi's PIR in the setting of $L = 2$ and $t \in \{1, 2, 4\}$

Specifically, we define the notion of *PIR with a trusted setup*, in which a trusted party computes an augmented database $\widetilde{D}$ and a digest $\mathsf{dig}$, which is a small piece of information related to $D$. The trusted party distributes $\widetilde{D}$ among servers and sends $\mathsf{dig}$ to a client. Jumping ahead, in our construction, $\widetilde{D}$ augments the original database $D$ with proofs or signatures guaranteeing the authenticity of contents and $\mathsf{dig}$ is a commitment to $D$ or a verification key used to verify the authenticity. After the setup, the client and servers execute an LD-PIR protocol as usual except that servers compute answers based on the augmented database $\widetilde{D}$ instead of $D$. The client computes an output from servers' answers and verifies its validity by using $\mathsf{dig}$. The formal definition of PIR with a trusted setup is shown in Appendix G.1.

## 9.1 Construction Based on Digital Signature

We construct a PIR protocol with a trusted setup such that the client can obtain the unique correct result even if more than $m/2$ servers inject errors, assuming an LD-PIR protocol $\Pi$ and a digital signature scheme $\Sigma$. Our construction only incurs small overheads: The resulting protocol has only $|\sigma|$ times larger communication complexity than $\Pi$, where $|\sigma|$ is the signature size of $\Sigma$; The size of a digest $\mathsf{dig}$ that the client receives is equal to the size of a verification key of $\Sigma$. Roughly speaking, in the setup, a trusted party who owns a database $D$ computes a signature $\sigma_x$ for a message $(D_x, x)$ for each $x$, and sends an augmented database $\widetilde{D} = (D_x, \sigma_x)_{x\in[N]}$ to servers and a verification key $\mathsf{pk}$ to the client. After executing the LD-PIR protocol $\Pi$ as usual, the client obtains a list $\mathcal{L}_0$ consisting of at most $L$ message-signature pairs $(y, \sigma)$. The client then checks for the validity of each message-signature pair $((y, x), \sigma)$ using $\mathsf{pk}$ and outputs a list $\mathcal{L}$ of the $y$'s whose corresponding signature passes verification. The resulting protocol achieves unique decoding (i.e., $\mathcal{L} = \{D_x\}$) since the list-decoding property of $\Pi$ guarantees that $D_x \in \mathcal{L}$ and the unforgeability of $\Sigma$ guarantees that any element $y \neq D_x$ is excluded in the verification phase. The formal descriptions are given in Appendix G.2.

## 9.2 Construction Based on Vector Commitment

The construction based on signature schemes work well in the static case (i.e., a database is never updated). However, when even a single element of the database is updated, the trusted party needs to perform the setup again since all the previous signatures should be revoked, which results in $O(N)$ communication between the trusted party and servers. We propose a PIR protocol with a trusted setup that allows for efficient database updates, based on vector commitments [9, 41]. Roughly speaking, in the setup, the trusted party who owns a database $D$ sends the client a commitment $C$ to $D$ as a digest; He also computes a proof $\pi_x$ for each $x \in [N]$ that $D_x$ is the $x$-th committed message with respect to $C$ and sends an augmented database $\widetilde{D} = (D_x, \pi_x)_{x\in[N]}$ to servers. Whenever $D$ is updated to $D'$, the trusted party only needs to send a small piece of update information $\mathsf{upd}$ of size $\ll N$ to servers, from which servers obtain a new augmented database $\widetilde{D}'$; He sends the client a new commitment $C'$ to $D'$. The unique decoding property of the protocol follows from the position binding property of the underlying vector commitment scheme. It is known that vector commitment schemes with efficient updates can be realized based on standard assumptions such as collision-resistant hash (or Merkle tree), RSA, and computational Diffie-Hellman [9, 41]. The formal descriptions of our construction are given in Appendix G.3.

**Remark 9.1.** If we use a computationally secure 1-server PIR protocol, then we can obtain a computationally secure $(b+1)$-server PIR protocol with a trusted setup achieving unique decoding against $b$ corrupted servers. Indeed, after a database is signed or committed in the setup, a client executes the 1-server protocol with each server independently and outputs a value that passes verification. Clearly, the client can reject an incorrect answer by a corrupted server and uniquely recover the correct value from an answer by an honest server. The advantage of using LD-PIR is that resulting protocols can be based on weaker cryptographic primitives not known to imply 1-server PIR, such as digital signatures or collision-resistant hash functions. In fact, according to the results of [16], it is supposedly impossible to obtain 1-server PIR from one-way functions, which are sufficient for digital signatures [36]. Furthermore, 1-server PIR is not known to be obtained from collision-resistant hash functions while the other direction is known [29].

# References


[1] Ambainis, A.: Upper bound on the communication complexity of private information retrieval. In: Automata, Languages and Programming. pp. 401–407 (1997)

[2] Barkol, O., Ishai, Y., Weinreb, E.: On d-multiplicative secret sharing. Journal of Cryptology **23**(4), 580–593 (2010)

[3] Barkol, O., Ishai, Y., Weinreb, E.: On locally decodable codes, self-correctable codes, and t-private PIR. Algorithmica **58**(4), 831–859 (2010)

[4] Beimel, A., Ishai, Y., Kushilevitz, E., Raymond, J.F.: Breaking the $O(n^{1/(2k-1)})$ barrier for information-theoretic private information retrieval. In: The 43rd Annual IEEE Symposium on Foundations of Computer Science, 2002. Proceedings. pp. 261–270 (2002)

[5] Beimel, A., Ishai, Y.: Information-theoretic private information retrieval: A unified construction. In: Automata, Languages and Programming. pp. 912–926 (2001)

[6] Beimel, A., Stahl, Y.: Robust information-theoretic private information retrieval. Journal of Cryptology **20**(3), 295–321 (2007)

[7] Ben-David, S., Kalai, Y.T., Paneth, O.: Verifiable private information retrieval. In: Theory of Cryptography. pp. 3–32 (2022)

[8] de Castro, L., Lee, K.: VeriSimplePIR: Verifiability in SimplePIR at no online cost for honest servers. In: 33rd USENIX Security Symposium (USENIX Security 24). pp. 5931–5948 (2024)

[9] Catalano, D., Fiore, D.: Vector commitments and their applications. In: Public-Key Cryptography – PKC 2013. pp. 55–72 (2013)

[10] 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)

[11] Chor, B., Goldreich, O., Kushilevitz, E., Sudan, M.: Private information retrieval. Journal of the ACM **45**(6), 965–982 (1998)

[12] 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)

[13] Corrigan-Gibbs, H., Henzinger, A., Kogan, D.: Single-server private information retrieval with sublinear amortized time. In: Advances in Cryptology – EUROCRYPT 2022. pp. 3–33 (2022)

[14] Damgård, I., Larsen, K.G., Nielsen, J.B.: Communication lower bounds for statistically secure mpc, with or without preprocessing. In: Advances in Cryptology – CRYPTO 2019. pp. 61–84 (2019)

[15] Devet, C., Goldberg, I., Heninger, N.: Optimally robust private information retrieval. In: 21st USENIX Security Symposium (USENIX Security 12). pp. 269–283 (2012)

[16] Di Crescenzo, G., Malkin, T., Ostrovsky, R.: Single database private information retrieval implies oblivious transfer. In: Advances in Cryptology – EUROCRYPT 2000. pp. 122–138 (2000)

[17] Dietz, M., Tessaro, S.: Fully malicious authenticated PIR. In: Advances in Cryptology – CRYPTO 2024. pp. 113–147 (2024)

[18] Downey, R.G., Fellows, M.R.: Fixed-parameter tractability and completeness ii: On completeness for W[1]. Theoretical Computer Science **141**(1), 109–131 (1995)

[19] Dvir, Z., Gopi, S.: 2-server PIR with subpolynomial communication. Journal of the ACM **63**(4), 1–15 (2016)

[20] Efremenko, K.: 3-query locally decodable codes of subexponential length. SIAM Journal on Computing **41**(6), 1694–1703 (2012)

[21] Eriguchi, R., Kunihiro, N., Nuida, K.: A linear algebraic approach to strongly secure ramp secret sharing for general access structures. In: 2020 International Symposium on Information Theory and Its Applications (ISITA). pp. 427–431 (2020)

[22] Eriguchi, R., Kurosawa, K., Nuida, K.: Multi-server PIR with full error detection and limited error correction. In: 3rd Conference on Information-Theoretic Cryptography (ITC 2022). pp. 1:1–1:20 (2022)

[23] Eriguchi, R., Kurosawa, K., Nuida, K.: On the optimal communication complexity of error-correcting multi-server PIR. In: Theory of Cryptography. pp. 60–88 (2022)

[24] Eriguchi, R., Kurosawa, K., Nuida, K.: Efficient and generic methods to achieve active security in private information retrieval and more advanced database search. In: Advances in Cryptology – EUROCRYPT 2024. pp. 92–121 (2024)

[25] Falk, B., Mishra, P., Shtepel, M.: Malicious security for PIR (almost) for free. Cryptology ePrint Archive, Paper 2024/964 (2024), `https://eprint.iacr.org/2024/964`

[26] Goldberg, I.: Improving the robustness of private information retrieval. In: 2007 IEEE Symposium on Security and Privacy (SP'07). pp. 131–148 (2007)

[27] Guruswami, V., Sudan, M.: Improved decoding of reed-solomon and algebraic-geometric codes. In: Proceedings 39th Annual Symposium on Foundations of Computer Science (FOCS '98). pp. 28–37 (1998)

[28] Guruswami, V., Wang, C.: Optimal rate list decoding via derivative codes. In: Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques. pp. 593–604 (2011)

[29] Ishai, Y., Kushilevitz, E., Ostrovsky, R.: Sufficient conditions for collision-resistant hashing. In: Theory of Cryptography. pp. 445–456 (2005)

[30] Itoh, T., Suzuki, Y.: Improved constructions for query-efficient locally decodable codes of subexponential length. IEICE Transactions on Information and Systems **E93.D**(2), 263–270 (2010)

[31] Kopparty, S.: List-decoding multiplicity codes. Theory of Computing **11**(5), 149–182 (2015)

[32] 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)

[33] Kurosawa, K.: How to correct errors in multi-server PIR. In: Advances in Cryptology – ASIACRYPT 2019. pp. 564–574 (2019)

[34] 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 (1997)

[35] Mughees, M.H., Chen, H., Ren, L.: OnionPIR: Response efficient single-server PIR. In: Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security. pp. 2292–2306. CCS '21 (2021)

[36] Rompel, J.: One-way functions are necessary and sufficient for secure signatures. In: Proceedings of the Twenty-Second Annual ACM Symposium on Theory of Computing. pp. 387–394. STOC '90 (1990)

[37] Shoup, V.: New algorithms for finding irreducible polynomials over finite fields. In: 29th Annual Symposium on Foundations of Computer Science. pp. 283–290 (1988)

[38] Shparlinski, I.: Finite Fields: Theory and Computation, vol. 477. Springer Science & Business Media (2013)

[39] Spitzbart, A.: A generalization of Hermite's interpolation formula. The American Mathematical Monthly **67**(1), 42–46 (1960)

[40] Sudan, M.: List decoding: Algorithms and applications. In: Theoretical Computer Science: Exploring New Frontiers of Theoretical Informatics. pp. 25–41 (2000)

[41] Tas, E.N., Boneh, D.: Vector Commitments with Efficient Updates. In: 5th Conference on Advances in Financial Technologies (AFT 2023). vol. 282, pp. 29:1–29:23 (2023)

[42] Woodruff, D., Yekhanin, S.: A geometric approach to information-theoretic private information retrieval. SIAM Journal on Computing **37**(4), 1046–1056 (2007)

[43] Yekhanin, S.: Towards 3-query locally decodable codes of subexponential length. Journal of the ACM (JACM) **55**(1), 1–16 (2008)

[44] Zhang, L.F., Safavi-Naini, R.: Verifiable multi-server private information retrieval. In: Applied Cryptography and Network Security. pp. 62–79 (2014)

[45] Zhang, L.F., Wang, H.: Multi-server verifiable computation of low-degree polynomials. In: 2022 IEEE Symposium on Security and Privacy (SP). pp. 596–613 (2022)

[46] 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. ASIA CCS '22 (2022)

# A Definitions of Multi-round PIR and LD-PIR

First, we define regular PIR in the multi-round setting.

**Definition A.1.** *An $\ell$-round $m$-server $t$-private* PIR protocol *is a tuple of three polynomial-time algorithms $\Pi = (\mathsf{Query}, \mathsf{Answer}, \mathsf{Output})$, where:*

- $\mathsf{Query}(1^\lambda, x, \mathsf{st}^{(j-1)}, (\mathsf{ans}_i^{(j-1)})_{i\in[m]}) \to ((\mathsf{que}_i^{(j)})_{i\in[m]}, \mathsf{st}^{(j)})$*:* $\mathsf{Query}$ *is a randomized algorithm that takes an index $x \in [N]$, a state $\mathsf{st}^{(j-1)}$ and answers $(\mathsf{ans}_i^{(j-1)})_{i\in[m]}$ in round $j-1$ as input, and outputs queries $(\mathsf{que}_i^{(j)})_{i\in[m]}$ and a state $\mathsf{st}^{(j)}$ in round $j$, where we define $\mathsf{st}^{(0)}$, $\mathsf{ans}_i^{(0)}$ as the empty string;*
- $\mathsf{Answer}(D, \mathsf{que}_i^{(j)}) \to \mathsf{ans}_i^{(j)}$*:* $\mathsf{Answer}$ *is a deterministic algorithm that takes a database $D \in \mathcal{X}^N$ and a query $\mathsf{que}_i^{(j)}$ in round $j$ as input, and outputs an answer $\mathsf{ans}_i^{(j)}$ in round $j$;*
- $\mathsf{Output}(\mathsf{st}^{(\ell)}, (\mathsf{ans}_i^{(\ell)})_{i\in[m]}) \to y$*:* $\mathsf{Output}$ *is a deterministic algorithm that takes a state $\mathsf{st}^{(\ell)}$ and answers $(\mathsf{ans}_i^{(\ell)})_{i\in[m]}$ in round $\ell$ as input, and outputs $y \in \mathcal{X}$;*

*such that $\Pi$ satisfies the following property:*

**Correctness.** *For any $\lambda \in \mathbb{N}$, any $D = (D_1, \ldots, D_N) \in \mathcal{X}^N$ and any $x \in [N]$, it holds that*

$$\Pr\left[\mathsf{Output}(\mathsf{st}^{(\ell)}, (\mathsf{ans}_i^{(\ell)})_{i\in[m]})) = D_x\right] = 1,$$

*where*

$$((\mathsf{que}_i^{(j)})_{i\in[m]}, \mathsf{st}^{(j)}) \leftarrow \mathsf{Query}(1^\lambda, x, \mathsf{st}^{(j-1)}, (\mathsf{ans}_i^{(j-1)})_{i\in[m]}),$$
$$\mathsf{ans}_i^{(j)} \leftarrow \mathsf{Answer}(D, \mathsf{que}_i^{(j)})$$

*for all $j \in [\ell]$ and $i \in [m]$.*

**Privacy.** *For any stateful algorithm $\mathcal{A}$, the distributions of $\mathsf{Priv}^0_{\Pi,\mathcal{A}}(\lambda)$ and $\mathsf{Priv}^1_{\Pi,\mathcal{A}}(\lambda)$ are statistically indistinguishable, where for $\alpha \in \{0,1\}$, $\mathsf{Priv}^\alpha_{\Pi,\mathcal{A}}(\lambda)$ is the output $\alpha'$ of $\mathcal{A}$ in the following experiment $\mathrm{EXPT}^{\mathrm{Priv},\alpha}_{\Pi,\mathcal{A}}(\lambda)$:*

1. *$\mathcal{A}(1^\lambda)$ outputs $(x_0, x_1, D, T)$, where $x_0, x_1 \in [N]$, $D \in \mathcal{X}^N$ and $T \subseteq [m]$ is of size at most $t$.*
2. *For each $j = 1, 2, \ldots, \ell$,*
   (a) *Let $((\mathsf{que}_i^{(j)})_{i\in[m]}, \mathsf{st}^{(j)}) \leftarrow \mathsf{Query}(1^\lambda, x_\alpha, \mathsf{st}^{(j-1)}, (\mathsf{ans}_i^{(j-1)})_{i\in[m]})$ and give $(\mathsf{que}_i^{(j)})_{i\in T}$ to $\mathcal{A}$.*
   (b) *If $j < \ell$, $\mathcal{A}$ outputs $(\mathsf{ans}_i^{(j)})_{i\in T}$. Let $\mathsf{ans}_i^{(j)} \leftarrow \mathsf{Answer}(D, \mathsf{que}_i^{(j)})$ for $i \notin T$.*
   (c) *If $j = \ell$, $\mathcal{A}$ outputs a bit $\alpha' \in \{0,1\}$.*

The communication complexity of $\Pi$ is defined as

$$\mathrm{Comm}(\Pi) = \max_{D\in\mathcal{X}^N, x\in[N]} \sum_{i\in[m], j\in[\ell]} (|\mathsf{que}_i^{(j)}| + |\mathsf{ans}_i^{(j)}|),$$

where $|\mathsf{que}_i^{(j)}|$ and $|\mathsf{ans}_i^{(j)}|$ are the bit-lengths of $\mathsf{que}_i^{(j)}$ and $\mathsf{ans}_i^{(j)}$, respectively. The computation complexity of $\Pi$ is defined as the total running time of $\mathsf{Query}$, $\mathsf{Answer}$ and $\mathsf{Output}$ with worst-case inputs $(D, x)$.

Next, we define LD-PIR in the multi-round setting.

**Definition A.2.** *Let $\Pi = (\mathsf{Query}, \mathsf{Answer}, \mathsf{Output})$ be an $\ell$-round $m$-server PIR protocol. We say that $\Pi$ is* $(b, L)$-list-decodable *if* $\mathsf{Output}$ *is allowed to output a subset $\mathcal{L} \subseteq \mathcal{X}$ and $\Pi$ satisfies the following requirements:*

**List decoding.** *There exists a negligible function $\epsilon = \epsilon(\lambda)$ such that for any $\lambda \in \mathbb{N}$ and any stateful algorithm $\mathcal{A}$, it holds that*

$$\Pr[\mathsf{LD}_{\Pi,\mathcal{A}}(\lambda) = 1] \le \epsilon,$$

*where $\mathsf{LD}_{\Pi,\mathcal{A}}(\lambda)$ is the output of the following experiment $\mathrm{EXPT}^{\mathrm{LD}}_{\Pi,\mathcal{A}}(\lambda)$:*

1. $\mathcal{A}(1^\lambda)$ *outputs* $(x, D, T)$*, where* $D \in \mathcal{X}^N$*,* $x \in [N]$ *and* $T \subseteq [m]$ *is of size at most* $b$*.*
2. *For each* $j = 1, 2, \ldots, \ell$*,*
   (a) *Let* $((\mathsf{que}_i^{(j)})_{i\in[m]}, \mathsf{st}^{(j)}) \leftarrow \mathsf{Query}(1^\lambda, x, \mathsf{st}^{(j-1)}, (\mathsf{ans}_i^{(j-1)})_{i\in[m]})$ *and give* $(\mathsf{que}_i^{(j)})_{i\in T}$ *to* $\mathcal{A}$*.*
   (b) $\mathcal{A}$ *outputs* $(\mathsf{ans}_i^{(j)})_{i\in T}$*. Let* $\mathsf{ans}_i^{(j)} \leftarrow \mathsf{Answer}(D, \mathsf{que}_i^{(j)})$ *for* $i \notin T$*.*
3. *Let* $\mathcal{L} \leftarrow \mathsf{Output}(\mathsf{st}^{(\ell)}, (\mathsf{ans}_i^{(\ell)})_{i\in[m]})$*. Return* 0 *if* $D_x \in \mathcal{L}$ *and* $|\mathcal{L}| \le L$*, and otherwise return* 1*.*

# B The Converse of Theorem 4.2

We can obtain an (inefficient) $(b, L)$-LD-PIR protocol if the list decoding bound is satisfied.

**Proposition B.1.** *Assume that* $m > b(L+1)/L$*. If there exists* 1*-private* $\mathsf{negl}(\lambda)$*-correct* 1*-server PIR protocol* $\Pi$*, then there exists an* $m$*-private* $m$*-server* $(b, L)$*-LD-PIR protocol* $\Pi'$ *such that* $\mathrm{Comm}(\Pi') = \mathrm{Comm}(\Pi) \cdot m$*.*

*Proof.* Let $\Pi'$ be a protocol in which a client executes $\Pi$ with each server independently. Let $D = (D_1, \ldots, D_N)$ be the database and $x$ be the client's index. Let $D_x^{(i)}$ be an element obtained after executing $\Pi$ with a server $\mathsf{S}_i$ for $i \in [m]$. Note that if $\mathsf{S}_i$ is honest, it holds that $D_x^{(i)} = D_x$ except with negligible probability $\mathsf{negl}(\lambda)$. The client then outputs a list $\mathcal{L} = \{y_1, \ldots, y_L\}$ which is obtained by choosing the $L$ most frequent elements in a multiset $\{D_x^{(1)}, \ldots, D_x^{(m)}\}$. If there is a tie, i.e., $y \neq y' \in \mathcal{X}$ appear the same number of times, the client chooses a smaller one according to some order in $\mathcal{X}$. Clearly, the size of $\mathcal{L}$ is at most $L$. To see that $D_x \in \mathcal{L}$, assume otherwise that $D_x \neq y_j$ for all $j \in [L]$. Let $H = \{i \in [m] : D_x^{(i)} = D_x\}$. Since at least $m - b$ servers are honest, we have that $|H| \ge m - b$ except with probability $(m-b)\mathsf{negl}(\lambda)$, which is still negligible. Thus, if $D_x \notin \mathcal{L}$, each $y_j$ must appear in $\{D_x^{(1)}, \ldots, D_x^{(m)}\}$ at least $m - b$ times. That is, there exist subsets $T_1, \ldots, T_L \subseteq [m]$ such that $H, T_1, \ldots, T_L$ are pairwise disjoint and $|\{i \in T_j : D_x^{(i)} = y_j\}| \ge m - b$ for all $j \in [L]$. However, this implies that $m \ge |T_1| + \cdots + |T_L| + |H| \ge L(m-b) + (m-b) = (L+1)(m-b)$, which yields $m \le b(L+1)/L$. It contradicts the assumption $m > b(L+1)/L$. We therefore conclude that $\Pi'$ is $(b, L)$-list-decodable. The privacy of $\Pi'$ is implied by that of $\Pi$. □

# C Proof of Theorem 7.1

Let $\kappa \in \mathbb{N}$ be the smallest integer such that $\kappa \ge 2bm(\ln\lambda)^2$. Let $\Pi_0$ be a sub-protocol described in Figure 3. Let $\Pi'$ be a protocol that executes $\Pi_0$ $\kappa$ times independently. Formally, $\Pi'$ is described in Figure 4. Correctness is straightforward. If all servers behave honestly, then $b_i^{(u)} = b_j^{(u)} = \mathsf{Answer}(D, \mathsf{que}_{\ell^{(u)}}^{(u)})$ for all $i, j \in [m]$ and $u \in [\kappa]$, where $\mathsf{que}_\ell^{(u)}$ is the query for a server $\mathsf{S}_\ell$ generated at the $u$-th instance of $\Pi_0$. Thus, all servers are included in the unique equivalence class $C_1$. Then, $y_1^{(u)}$ is equal to $D_x$ for all $u \in [\kappa]$ and we have $\mathcal{L} = \{D_x\}$. Note that $\Pi_0$ is $t$-private even if servers behave maliciously since any $t$ servers obtain at most $t+1$ queries of the $(t+1)$-private protocol $\Pi$. Since $\Pi'$ simply executes $\Pi_0$ in parallel, $\Pi'$ also satisfies $t$-privacy. The communication complexity of $\Pi'$ is $\kappa$ times larger than that of $\Pi_0$, which is at most twice larger than that of $\Pi$. Hence we have $\mathrm{Comm}(\Pi') = \mathrm{Comm}(\Pi) \cdot \widetilde{O}(mb)$. Since the most costly step of $\Pi'$ is to partition the set of servers into equivalence classes, which can be done in time $O(m^2)$, the computation complexity of $\Pi'$ is also $O(\kappa) = \widetilde{O}(m^2)$ times larger than that of $\Pi_0$.

We show that $\Pi'$ is $(b, L)$-list-decodable. Let $H$ be the set of all $m - b$ honest servers and $B = \overline{H}$. Any pair of honest servers $\mathsf{S}_i, \mathsf{S}_j \in H$ are equivalent under the equivalence relation defined as Step 3 of $\Pi'$. This is because $b_i^{(u)} = b_j^{(u)} = \mathsf{Answer}(D, \mathsf{que}_{\ell^{(u)}}^{(u)})$. Therefore, there exists $g \in [h]$ such that $H \subseteq C_g$ at Step 4 of $\Pi'$. In addition, we can see that $h \le L$. Indeed, if $h \ge L+1$, we would have that $(m-b)(L+1) \le m$ and hence $m \le b(L+1)/L$, which contradicts the assumption $m > b(L+1)/L$.

Let $\mathsf{F}$ be the event that $\mathsf{C}$ outputs a list $\mathcal{L}$ such that $D_x \notin \mathcal{L}$. Note that the size of $\mathcal{L}$ is at most $L$ with probability 1 since $h \le L$. For $i \in [m]$ with $\mathsf{S}_i \in B$, let $K_i \subseteq [\kappa]$ be the set of all $u$'s such that $a_i^{(u)} \neq \mathsf{Answer}(D, \mathsf{que}_i^{(u)})$. The event $\mathsf{F}$ happens only if there exists $B_g \subseteq B$ such that servers in $B$ are included in the same equivalence class $C_g$ as $H$. Furthermore, at least $\kappa/2$ values of $y_g^{(1)}, \ldots, y_g^{(\kappa)}$ are not

**Notations.**

- Let $k = m - b$.
- Let $\Pi = (\mathsf{Query}, \mathsf{Answer}, \mathsf{Output})$ be a $k$-robust $(t+1)$-private 1-round $m$-server PIR protocol.

**Input.** A client $\mathsf{C}$ has an index $x \in [N]$ and each server $\mathsf{S}_i$ has a database $D \in \mathcal{X}^N$.

**Output.** $\mathsf{C}$ obtains a tuple of answers $(a_i, b_i)_{i\in[m]}$ and a state $\mathsf{st}$.

**Protocol.**

**First round.**

1. $\mathsf{C}$ computes $(\mathsf{que}_1, \ldots, \mathsf{que}_m, \mathsf{st}) \leftarrow \mathsf{Query}(x)$.
2. $\mathsf{C}$ sends $\mathsf{que}_i$ to each server $\mathsf{S}_i$.
3. $\mathsf{S}_i$ sends $\mathsf{Answer}(D, \mathsf{que}_i)$ to $\mathsf{C}$.

**Second round.**

1. $\mathsf{C}$ chooses $\ell \leftarrow_{\$} [m]$.
2. $\mathsf{C}$ sends $\mathsf{que}_\ell$ to servers $\mathsf{S}_i$ for $i \in [m] \setminus \{\ell\}$.
3. For $i \in [m] \setminus \{\ell\}$, $\mathsf{S}_i$ sends $\mathsf{Answer}(D, \mathsf{que}_\ell)$ to $\mathsf{C}$.

**Output.**

1. For each $i \in [m]$, let $a_i$ and $b_i$ denote the answers returned by $\mathsf{S}_i$ at the first and second rounds, respectively, where we define $b_\ell := a_\ell$.
2. $\mathsf{C}$ obtains $(a_i, b_i)_{i\in[m]}$ and $\mathsf{st}$.

Figure 3: A sub-protocol $\Pi_0$ in the proof of Theorem 7.1

**Notations.** Continuing the notations in Figure 3,

- Let $\Pi_0$ be a sub-protocol described in Figure 3.
- Let $\kappa \in \mathbb{N}$ be a parameter.

**Input.** A client $\mathsf{C}$ has an index $x \in [N]$ and each server $\mathsf{S}_i$ has a database $D \in \mathcal{X}^N$.

**Output.** $\mathsf{C}$ obtains a list $\mathcal{L}$.

**Protocol.**

1. $\mathsf{C}$ executes $\kappa$ independent instances of $\Pi_0$ in parallel.
2. For $u \in [\kappa]$, let $\ell^{(u)}$ be the index chosen by $\mathsf{C}$ in the $u$-th instance of $\Pi_0$ and let $(a_i^{(u)}, b_i^{(u)})_{i\in[m]}$ and $\mathsf{st}^{(u)}$ be the outputs of the $u$-th instance of $\Pi_0$.
3. $\mathsf{C}$ partitions the set of servers into equivalence classes under the following equivalence relation:
$$\mathsf{S}_i \sim \mathsf{S}_j \overset{\text{def}}{\iff} b_i^{(u)} = b_j^{(u)} \ (\forall u \in [\kappa]).$$
4. Let $C_1, \ldots, C_h$ be all the equivalence classes of size at least $k = m - b$.
5. For each $g \in [h]$, $\mathsf{C}$ does the following:
   (a) $\mathsf{C}$ chooses a $k$-sized subset $C'_g$ of $C_g$ and computes
   $$y_g^{(u)} = \mathsf{Output}(C'_g, \mathsf{st}^{(u)}, (a_i^{(u)})_{i\in C'_g})$$
   for all $u \in [\kappa]$.
   (b) If there exists $y \in \mathcal{X}$ such that $|\{u \in [\kappa] : y_g^{(u)} = y\}| > \kappa/2$, then $\mathsf{C}$ sets $y_g := y$. Otherwise, $\mathsf{C}$ sets $y_g$ to the default value $y_0$.
6. $\mathsf{C}$ outputs a list $\mathcal{L} = \{y_g : g \in [h]\}$.

Figure 4: An LD-PIR protocol $\Pi'$ in Theorem 7.1

equal to $D_x$, and it thus holds that $|\bigcup_{i\in B_g} K_i| \geq \kappa/2$. Since $|B_g| \leq b$, there exists $\mathsf{S}_i \in B_g \subseteq \overline{H}$ such that $|K_i| \geq \kappa/(2b)$. For this server $\mathsf{S}_i$, it follows from $\mathsf{S}_i \in C_g$ that $\ell^{(u)} \neq i$ for all $u \in K_i$ since otherwise the answer $a_i^{(u)}$ is compared with the answers $b_j^{(u)}$ of honest servers $\mathsf{S}_j \in H$ and then $\mathsf{S}_i$ is not included in $B_g$. The probability that it happens is at most $(1-1/m)^{\kappa/(2b)} \leq \exp(-\kappa/(2bm))$. Therefore, it follows from the union bound that the probability that $\mathsf{F}$ happens is $b\exp(-\kappa/(2bm)) = \mathsf{negl}(\lambda)$.

# D Proof of Theorem 7.5

Let $\kappa \in \mathbb{N}$ be such that $\kappa = \Theta(mb(\ln\lambda)^2)$. Let $\Pi_0$ be a sub-protocol described in Figure 5. Let $\Pi'$ be a protocol that executes $\Pi_0$ $\kappa$ times independently. Formally, $\Pi'$ is described in Figure 6.

**Privacy.** $\Pi_0$ is $t$-private since any $t$ servers obtain at most $t$ queries from each independent execution of $\Pi$. Since $\Pi'$ simply executes $\Pi_0$ in parallel, $\Pi'$ also satisfies $t$-privacy.

**Correctness.** Assume that all servers are honest. We define some notations. For each $u \in [\kappa]$, let $(i^{(u)}, \alpha^{(u)})$ denote randomness sampled by $\mathsf{C}$ at Step 1 of **Query** of the $u$-th instance of $\Pi_0$. Let $\mathsf{que}_{b,i}^{(u)}$ and $\mathsf{st}_b^{(u)}$ denote a query $\mathsf{que}_{b,i}$ and a state $\mathsf{st}_b$ computed by $\mathsf{C}$ at Step 2 in the $u$-th instance of $\Pi_0$. Let $q_\ell^{(u)}$, $r_\ell^{(u)}$ denote $q_\ell$, $r_\ell$ computed by $\mathsf{C}$ in the $u$-th instance of $\Pi_0$, respectively. First, any pair of servers $\mathsf{S}_i, \mathsf{S}_j$ are equivalent under the equivalence relation defined at Step 3 of $\Pi'$. Indeed, for any $u \in [\kappa]$, it holds that $b_i^{(u)} = \mathsf{Answer}(D, r_{i^{(u)}}^{(u)}) = b_j^{(u)}$. In particular, we have that $h = 1$ and $C_1 = [m]$ is the unique equivalence class. At Step 5, for any $g \in [h]$ and any $u \in [\kappa]$, it holds that

$$\begin{aligned}
y_1^{(u)} &= \mathsf{Output}\left(C'_1, \mathsf{st}_{\alpha^{(u)}}^{(u)}, (a_\ell^{(u)})_{\ell\in C'_1}\right) \\
&= \mathsf{Output}\left(C'_1, \mathsf{st}_{\alpha^{(u)}}^{(u)}, (\mathsf{Answer}(D, q_\ell^{(u)}))_{\ell\in C'_1}\right) \\
&= \mathsf{Output}\left(C'_1, \mathsf{st}_{\alpha^{(u)}}^{(u)}, (\mathsf{Answer}(D, \mathsf{que}_{\alpha^{(u)},\ell}^{(u)}))_{\ell\in C'_1}\right) \\
&= D_x.
\end{aligned}$$

Therefore, $y_1 = D_x$ and $\mathsf{C}$ outputs $\mathcal{L} = \{D_x\}$.

**List Decoding.** We show that $\Pi'$ is $(b, L)$-list-decodable. Let $H$ be the set of all $m-b$ honest servers and $B = \overline{H}$. Any pair of honest servers $\mathsf{S}_i, \mathsf{S}_j \in H$ are equivalent under the equivalence relation defined at Step 3 of $\Pi'$ since $b_i^{(u)} = \mathsf{Answer}(D, r_{i^{(u)}}^{(u)}) = b_j^{(u)}$ for any $u \in [\kappa]$. Therefore, there exists $g \in [h]$ such that $H \subseteq C_g$ at Step 4 of $\Pi'$. For simplicity, we assume that $H \subseteq C_1$. In addition, we can see that $h \leq L$ and hence the size of $\mathcal{L}$ outputted by $\mathsf{C}$ is at most $L$ with probability 1. Indeed, if $h \geq L+1$, we would have that $(m-b)(L+1) \leq m$ and hence $m \leq b(L+1)/L$, which contradicts the assumption $m > b(L+1)/L$.

Let $\mathsf{F}$ be the event that $\mathsf{C}$ outputs a list $\mathcal{L}$ such that $D_x \notin \mathcal{L}$. Let $R$ be a random variable representing the client's randomness at Step 2 of **Query** of $\Pi_0$ and the adversary's randomness to compute responses in **Answer** of $\Pi_0$. It is sufficient to show that $\Pr[\mathsf{F} \mid R = \rho]$ is negligible for any fixed value $\rho$ since $\Pr[\mathsf{F}] = \sum_\rho \Pr[R=\rho]\Pr[\mathsf{F} \mid R = \rho]$. In the following, we fix $R = \rho$ and omit it from descriptions for clarity.

For $i \in B$, we define $K_i = \{u \in [\kappa] : a_i^{(u)} = \mathsf{Answer}(D, q_i^{(u)})\}$. Assume that the event $\mathsf{F}$ happens. Then, there exists $B_1 \subseteq B$ such that servers in $B_1$ are included in the same equivalence class $C_1$ as $H$. Furthermore, at least $\kappa/2$ values of $y_1^{(1)}, \ldots, y_1^{(\kappa)}$ should be different from $D_x$, which implies that $|\bigcup_{i\in B_1} K_i| \geq \kappa/2$. Since $|B_1| \leq b$, there exists $i \in B_1$ such that the following event occurs:

$$\mathsf{E}_i : \ |K_i| \geq \frac{\kappa}{2b}$$

On the other hand, since $i \in C_1$, for any $u \in K_i$, the following event occurs:

$$\mathsf{F}_i^{(u)} : \ i^{(u)} \neq i \vee b_i^{(u)} = \mathsf{Answer}(D, r_i^{(u)})$$

since otherwise the answer $b_i^{(u)}$ from $\mathsf{S}_i$ is compared with the answer $b_j^{(u)} = \mathsf{Answer}(D, r_i^{(u)})$ from each honest server $\mathsf{S}_j$ and then $\mathsf{S}_i$ cannot be included the equivalence class $C_1$. We define $\mathsf{F}_i$ as the event that $\mathsf{F}_i^{(u)}$ occurs for all $u \in K_i$.

**Notations.**

- Let $k = m - b$.
- Let $\Pi = (\mathsf{Query}, \mathsf{Answer}, \mathsf{Output})$ be a $k$-robust $t$-private 1-round $m$-server PIR protocol.

**Input.** A client $\mathsf{C}$ has an index $x \in [N]$ and each server $\mathsf{S}_i$ has a database $D \in \mathcal{X}^N$.

**Output.** $\mathsf{C}$ obtains two tuples of answers $\boldsymbol{a} = (a_i)_{i\in[m]}$, $\boldsymbol{b} = (b_i)_{i\in[m]}$, and a state $\mathsf{st}'$.

**Protocol.**

**Query.**

1. $\mathsf{C}$ chooses $i \leftarrow_{\$} [m]$ and $\alpha \leftarrow_{\$} \{0,1\}$.
2. $\mathsf{C}$ computes

$$(\mathsf{que}_{0,1}, \ldots, \mathsf{que}_{0,m}, \mathsf{st}_0) \leftarrow \mathsf{Query}(x),\ (\mathsf{que}_{1,1}, \ldots, \mathsf{que}_{1,m}, \mathsf{st}_1) \leftarrow \mathsf{Query}(x)$$

   based on independent randomness.
3. $\mathsf{C}$ does the following:
   - If $\alpha = 0$, then $\mathsf{C}$ sets $\boldsymbol{q} = (q_\ell)_{\ell\in[m]}$, $\boldsymbol{r} = (r_\ell)_{\ell\in[m]}$ as $q_\ell = \mathsf{que}_{0,\ell}$ and $r_\ell = \mathsf{que}_{1,i}$ for all $\ell \in [m]$, that is,

$$\boldsymbol{q} = (\mathsf{que}_{0,1}, \ldots, \mathsf{que}_{0,m}),\ \boldsymbol{r} = (\mathsf{que}_{1,i}, \ldots, \mathsf{que}_{1,i}).$$

     $\mathsf{C}$ sends $(q_\ell, r_\ell)$ to $\mathsf{S}_\ell$.
   - If $\alpha = 1$, then $\mathsf{C}$ sets

$$\boldsymbol{r} = (\mathsf{que}_{0,i}, \ldots, \mathsf{que}_{0,i}),\ \boldsymbol{q} = (\mathsf{que}_{1,1}, \ldots, \mathsf{que}_{1,m}).$$

     $\mathsf{C}$ sends $(r_\ell, q_\ell)$ to $\mathsf{S}_\ell$.

**Answer.** Upon receiving two queries $(p_0, p_1)$, each server $\mathsf{S}_\ell$ sends $\mathsf{C}$

$$\mathsf{ans}_\ell = (\mathsf{Answer}(D, p_0), \mathsf{Answer}(D, p_1)).$$

**Output.**

1. Upon receiving $\mathsf{ans}_\ell$ from each $\mathsf{S}_\ell$, $\mathsf{C}$ sets $a_\ell, b_\ell$ as

$$\mathsf{ans}_\ell = (a_\ell, b_\ell) \text{ if } \alpha = 0, \text{ and } \mathsf{ans}_\ell = (b_\ell, a_\ell) \text{ if } \alpha = 1.$$

2. $\mathsf{C}$ outputs $\boldsymbol{a} = (a_1, \ldots, a_m)$, $\boldsymbol{b} = (b_1, \ldots, b_m)$ and $\widetilde{\mathsf{st}} = (\mathsf{st}_\alpha, i)$.

Figure 5: A sub-protocol $\Pi_0$ in the proof of Theorem 7.5

**Notations.** Continuing the notations in Figure 5,

- Let $\Pi_0$ be a sub-protocol described in Figure 5.
- Let $\kappa \in \mathbb{N}$ be a parameter.

**Input.** A client $\mathsf{C}$ has an index $x \in [N]$ and each server $\mathsf{S}_i$ has a database $D \in \mathcal{X}^N$.

**Output.** $\mathsf{C}$ obtains a list $\mathcal{L}$.

**Protocol.**

1. $\mathsf{C}$ executes $\kappa$ independent instances of $\Pi_0$ in parallel.
2. For $u \in [\kappa]$, let $\boldsymbol{a}^{(u)} = (a_\ell^{(u)})_{\ell \in [m]}$, $\boldsymbol{b}^{(u)} = (b_\ell^{(u)})_{\ell \in [m]}$ and $\widetilde{\mathsf{st}}^{(u)} = (\mathsf{st}^{(u)}, i^{(u)})$ be the outputs of the $u$-th instance of $\Pi_0$.
3. $\mathsf{C}$ partitions the set of servers into equivalence classes under the following equivalence relations:
$$\mathsf{S}_i \sim \mathsf{S}_j \overset{\text{def}}{\iff} b_i^{(u)} = b_j^{(u)} \ (\forall u \in [\kappa]).$$
4. Let $C_1, \ldots, C_h$ be all the equivalence classes of size at least $k$.
5. For each $g \in [h]$, $\mathsf{C}$ does the following:
   (a) $\mathsf{C}$ chooses a $k$-sized subset $C'_g$ of $C_g$ and computes
   $$y_g^{(u)} = \mathsf{Output}(C'_g, \mathsf{st}^{(u)}, (a_\ell^{(u)})_{\ell \in C'_g})$$
   for all $u \in [\kappa]$.
   (b) If there exists $y \in \mathcal{X}$ such that $|\{u \in [\kappa] : y_g^{(u)} = y\}| > \kappa/2$, then $\mathsf{C}$ sets $y_g = y$. Otherwise, $\mathsf{C}$ sets $y_g$ to the default value $y_0$.
6. $\mathsf{C}$ outputs a list $\mathcal{L} = \{y_g : g \in [h]\}$.

Figure 6: An LD-PIR protocol $\Pi'$ in Theorem 7.5

From the above observation, the event $\mathsf{F}$ happens only if there exist $i \in B$ such that $\mathsf{E}_i$ and $\mathsf{F}_i$ occur. We have that

$$\begin{aligned}\Pr[\mathsf{F}] &\leq \Pr[\exists i \in B : \mathsf{E}_i \wedge \mathsf{F}_i] \\ &\leq \sum_{i \in B} \Pr[\mathsf{E}_i \wedge \mathsf{F}_i] \\ &\leq \sum_{i \in B} \sum_{A : |A| \geq \frac{\kappa}{2b}} \Pr[K_i = A] \Pr[\mathsf{F}_i \mid K_i = A]. \end{aligned} \tag{5}$$

For $i \in B$ and $A$ with $|A| \geq \kappa/(2b)$, we analyze $\Pr[\mathsf{F}_i \mid K_i = A]$. For simplicity, we assume that $A = \{1, 2, \ldots, v\}$, where $v = |A|$. We have that

$$\Pr[\mathsf{F}_i \mid K_i = A] = \prod_{u=1}^{v} \Pr\left[\mathsf{F}_i^{(u)} \;\middle|\; (K_i = A) \wedge \mathsf{F}_i^{(1)} \wedge \cdots \wedge \mathsf{F}_i^{(u-1)}\right]. \tag{6}$$

Let $\pi^{(u)} = (i^{(u)}, \alpha^{(u)})$ denote randomness sampled by $\mathsf{C}$ at Step 1 of **Query** of the $u$-th instance of $\Pi_0$. Then,

$$\begin{aligned}&\Pr\left[\mathsf{F}_i^{(u)} \;\middle|\; (K_i = A) \wedge \mathsf{F}_i^{(1)} \wedge \cdots \wedge \mathsf{F}_i^{(u-1)}\right] \\ = &\sum_{\substack{\pi^{(1)}, \ldots, \pi^{(u-1)}, \\ \pi^{(u+1)}, \ldots, \pi^{(v)}}} \Pr\left[\pi^{(1)}, \ldots, \pi^{(u-1)}, \pi^{(u+1)}, \ldots, \pi^{(v)}\right] \\ &\times \Pr_{\pi^{(u)}}\left[\mathsf{F}_i^{(u)} \;\middle|\; (K_i = A) \wedge \mathsf{F}_i^{(1)} \wedge \cdots \wedge \mathsf{F}_i^{(u-1)}, \pi^{(1)}, \ldots, \pi^{(u-1)}, \pi^{(u+1)}, \ldots, \pi^{(v)}\right].\end{aligned} \tag{7}$$

Fix $\pi^{(1)}, \ldots, \pi^{(u-1)}, \pi^{(u+1)}, \ldots, \pi^{(v)}$. For ease of reading, let $\mathsf{COND}$ denote the condition of the probability (7). Define $\mathsf{BAD}$ be an event that $\mathsf{C}$ picks $\pi^{(u)} = (i^{(u)}, \alpha^{(u)})$ such that $i^{(u)} \neq i$. Then, we have that

$$\begin{aligned}&\Pr_{\pi^{(u)}}\left[\mathsf{F}_i^{(u)} \;\middle|\; \mathsf{COND}\right] \\ \leq\ &\Pr[\mathsf{BAD}] + \Pr\left[\overline{\mathsf{BAD}}\right] \cdot \Pr_{\pi^{(u)}}\left[\mathsf{F}_i^{(u)} \;\middle|\; \mathsf{COND} \wedge \overline{\mathsf{BAD}}\right] \\ \leq\ &\frac{m-1}{m} + \frac{1}{m} \cdot \Pr_{\pi^{(u)}}\left[\mathsf{F}_i^{(u)} \;\middle|\; \mathsf{COND} \wedge \overline{\mathsf{BAD}}\right].\end{aligned} \tag{8}$$

Since all randomness except for $\alpha^{(u)}$ is fixed, queries sent to servers in the $u'$-th instance for $u' \neq u$ are the same and those in the $u$-th instance are as follows:

$$\begin{array}{|c|c|}\hline \mathsf{que}_{0,1}^{(u)} & \mathsf{que}_{1,i}^{(u)} \\ \hline \vdots & \vdots \\ \hline \mathsf{que}_{0,i}^{(u)} & \mathsf{que}_{1,i}^{(u)} \\ \hline \vdots & \vdots \\ \hline \mathsf{que}_{0,m}^{(u)} & \mathsf{que}_{1,i}^{(u)} \\ \hline \end{array} \text{ if } \alpha^{(u)} = 0, \text{ or } \begin{array}{|c|c|}\hline \mathsf{que}_{0,i}^{(u)} & \mathsf{que}_{1,1}^{(u)} \\ \hline \vdots & \vdots \\ \hline \mathsf{que}_{0,i}^{(u)} & \mathsf{que}_{1,i}^{(u)} \\ \hline \vdots & \vdots \\ \hline \mathsf{que}_{0,i}^{(u)} & \mathsf{que}_{1,m}^{(u)} \\ \hline \end{array} \text{ if } \alpha^{(u)} = 1.$$

In particular, queries seen by $\mathsf{S}_i$ are the same regardless of the value of $\alpha^{(u)}$. Since we assume the 1-local error model, the answer $b_i^{(u)}$ from $\mathsf{S}_i$ is a function of $(\mathsf{que}_{0,i}^{(u')}, \mathsf{que}_{1,i}^{(u')})_{u' \in [\kappa]}$. Therefore, $b_i^{(u)}$ are also independent of $\alpha^{(u)}$ and it holds that

$$\Pr_{\pi^{(u)}}\left[\mathsf{F}_i^{(u)} \;\middle|\; \mathsf{COND} \wedge \overline{\mathsf{BAD}}\right] \leq \frac{1}{2}. \tag{9}$$

From (8), we have that

$$\Pr_{\pi^{(u)}}\left[\mathsf{F}_{ij}^{(u)} \;\middle|\; \mathsf{COND}\right] \leq 1 - \frac{1}{2m}.$$

Combining (5), (6), and (7), we conclude that the probability of $\mathsf{F}$ is upper bounded by $|B|(1-1/(2m))^{\kappa/(2b)} \leq b\exp(-\kappa/(4mb)) \leq \exp(-\Omega((\ln\lambda)^2))$, which is negligible in $\lambda$.

**Complexity.** The communication complexity of $\Pi'$ is $\kappa$ times larger than that of $\Pi_0$, which is at most twice larger than that of $\Pi$. Hence, we have $\mathrm{Comm}(\Pi') = \mathrm{Comm}(\Pi)\cdot 2\kappa = \mathrm{Comm}(\Pi)\cdot\widetilde{O}(mb)$. All the steps except for Step 3 of $\Pi'$ can be done in time $O(\kappa)\cdot\mathrm{Comp}(\Pi) = \widetilde{O}(mb)\cdot\mathrm{Comp}(\Pi)$. The computation complexity of Step 3 of $\Pi'$ is at most $O(m^2)$. Thus, we have $\mathrm{Comp}(\Pi') = \widetilde{O}(m^2)\cdot\mathrm{Comp}(\Pi)$.

# E Proof of Theorem 7.8

Let $\kappa \in \mathbb{N}$ be such that $\kappa = \Theta(m^2 b(\ln\lambda)^2)$. Let $\Pi_0$ be a sub-protocol described in Figure 7. Let $\Pi'$ be a protocol that executes $\Pi_0$ $\kappa$ times independently. Formally, $\Pi'$ is described in Figure 8.

**Privacy.** $\Pi_0$ is $t$-private since any $t$ servers obtain at most $t$ queries from each independent execution of $\Pi$. Since $\Pi'$ simply executes $\Pi_0$ in parallel, $\Pi'$ also satisfies $t$-privacy.

**Correctness.** Assume that all servers are honest. We define some notations. For each $u \in [\kappa]$, let $(i^{(u)}, j^{(u)}, \alpha^{(u)})$ denote randomness sampled by $\mathsf{C}$ at Step 1 of **Query** of the $u$-th instance of $\Pi_0$. Let $\mathsf{que}_{b,i}^{(u)}$ and $\mathsf{st}_b^{(u)}$ denote a query $\mathsf{que}_{b,i}$ and a state $\mathsf{st}_b$ computed by $\mathsf{C}$ at Step 2 in the $u$-th instance of $\Pi_0$. Let $q_\ell^{(u)}$, $r_\ell^{(u)}$ denote $q_\ell$, $r_\ell$ computed by $\mathsf{C}$ in the $u$-th instance of $\Pi_0$, respectively. First, the graph $G$ computed at Step 3 of $\Pi'$ remains a complete graph. Indeed, for any $u \in [\kappa]$, it holds that $b_{i^{(u)}}^{(u)} = \mathsf{Answer}(D, r_{i^{(u)}}^{(u)})$ and $b_{j^{(u)}}^{(u)} = \mathsf{Answer}(D, r_{j^{(u)}}^{(u)})$. Since $r_{j^{(u)}}^{(u)} = r_{i^{(u)}}^{(u)} = \mathsf{que}_{1-\alpha^{(u)},j^{(u)}}^{(u)}$, we have $b_{i^{(u)}}^{(u)} = b_{j^{(u)}}^{(u)}$ and hence an edge $(i,j)$ is never removed. In particular, we have that at Step 5, for any $g \in [h]$ and any $u \in [\kappa]$,

$$\begin{aligned} y_g^{(u)} &= \mathsf{Output}\left(C_g, \mathsf{st}_{\alpha^{(u)}}^{(u)}, (a_\ell^{(u)})_{\ell\in C_g}\right) \\ &= \mathsf{Output}\left(C_g, \mathsf{st}_{\alpha^{(u)}}^{(u)}, (\mathsf{Answer}(D, q_\ell^{(u)}))_{\ell\in C_g}\right) \\ &= \mathsf{Output}\left(C_g, \mathsf{st}_{\alpha^{(u)}}^{(u)}, (\mathsf{Answer}(D, \mathsf{que}_{\alpha^{(u)},\ell}^{(u)}))_{\ell\in C_g}\right) \\ &= D_x. \end{aligned}$$

Therefore, $y_g = D_x$ for any $g \in [h]$ and $\mathsf{C}$ outputs $\mathcal{L} = \{D_x\}$.

**List Decoding.** We show that $\Pi'$ is $(b, L)$-list-decodable. Let $H$ be the set of all $m-b$ honest servers and $B = \overline{H}$. We see that $H$ forms a clique in the graph $G$ computed by $\mathsf{C}$ at Step 3 of $\Pi'$ and hence at least one clique of size $k$ is found at Step 4. Indeed, if there does not exist an edge $(i,j)$ between honest servers $\mathsf{S}_i, \mathsf{S}_j \in H$ in $G$, then it implies that there exists $u \in [\kappa]$ such that (1) $i^{(u)} = i$, $j^{(u)} = j$, and $b_{i^{(u)}}^{(u)} \neq b_{j^{(u)}}^{(u)}$, or (2) $i^{(u)} = j$, $j^{(u)} = i$, and $b_{i^{(u)}}^{(u)} \neq b_{j^{(u)}}^{(u)}$. However, since $\mathsf{S}_i, \mathsf{S}_j$ are honest, we have that $b_i^{(u)} = \mathsf{Answer}(D, r_i^{(u)}) = b_j^{(u)}$ in the first case and $b_j^{(u)} = \mathsf{Answer}(D, r_j^{(u)}) = b_i^{(u)}$ in the second case, which leads to a contradiction. In addition, we can see that $h \leq L$ and hence the size of $\mathcal{L}$ outputted by $\mathsf{C}$ is at most $L$ with probability 1. Indeed, if $h \geq L+1$, we would have that $(m-b)(L+1) \leq m$ and hence $m \leq b(L+1)/L$, which contradicts the assumption $m > b(L+1)/L$.

Let $\mathsf{F}$ be the event that $\mathsf{C}$ outputs a list $\mathcal{L}$ such that $D_x \notin \mathcal{L}$. Let $R$ be a random variable representing the client's randomness at Step 2 of **Query** of $\Pi_0$ and the adversary's randomness to compute responses in **Answer** of $\Pi_0$. It is sufficient to show that $\Pr[\mathsf{F} \mid R = \rho]$ is negligible for any fixed value $\rho$ since $\Pr[\mathsf{F}] = \sum_\rho \Pr[R=\rho]\Pr[\mathsf{F} \mid R = \rho]$. In the following, we fix $R = \rho$ and omit it from descriptions for clarity.

For $i \in B$, we define $K_i = \{u \in [\kappa] : a_i^{(u)} = \mathsf{Answer}(D, q_i^{(u)})\}$. Assume that the event $\mathsf{F}$ happens. Then, there exist two honest servers $j, j' \in H$ such that $j$ and $j'$ are included in distinct cliques, i.e., $i \in C_g$ and $j \in C_{g'}$ for some $g \neq g'$, since otherwise we have that $H = C_g$ for some $g$ and $D_x = y_g \in \mathcal{L}$. Let $B_g = C_g \setminus H$. Furthermore, at least $\kappa/2$ values of $y_g^{(1)}, \ldots, y_g^{(\kappa)}$ should be different from $D_x$, which implies that $|\bigcup_{i\in B_g} K_i| \geq \kappa/2$. Since $|B_g| \leq b$, there exists $i \in B_g$ such that the following event occurs:

$$\mathsf{E}_i : \ |K_i| \geq \frac{\kappa}{2b}$$

**Notations.**

- Let $k = m - b$.
- Let $\Pi = (\mathsf{Query}, \mathsf{Answer}, \mathsf{Output})$ be a $k$-robust $t$-private 1-round $m$-server PIR protocol.
- Let $\mathcal{I} = \{(i,j) \in [m]^2 : i \neq j\}$.

**Input.** A client $\mathsf{C}$ has an index $x \in [N]$ and each server $\mathsf{S}_i$ has a database $D \in \mathcal{X}^N$.

**Output.** $\mathsf{C}$ obtains two tuples of answers $\boldsymbol{a} = (a_i)_{i\in[m]}$, $\boldsymbol{b} = (b_i)_{i\in[m]}$, and a state $\mathsf{st}'$.

**Protocol.**

**Query.**

1. $\mathsf{C}$ chooses $(i,j) \leftarrow\!\!\$\, \mathcal{I}$ and $\alpha \leftarrow\!\!\$\, \{0,1\}$.
2. $\mathsf{C}$ computes

$$(\mathsf{que}_{0,1}, \ldots, \mathsf{que}_{0,m}, \mathsf{st}_0) \leftarrow \mathsf{Query}(x),\ (\mathsf{que}_{1,1}, \ldots, \mathsf{que}_{1,m}, \mathsf{st}_1) \leftarrow \mathsf{Query}(x)$$

   based on independent randomness.
3. $\mathsf{C}$ does the following:
   - If $\alpha = 0$, then $\mathsf{C}$ sets $\boldsymbol{q} = (q_\ell)_{\ell\in[m]}$, $\boldsymbol{r} = (r_\ell)_{\ell\in[m]}$ as

$$\begin{aligned}\boldsymbol{q} &= (\mathsf{que}_{0,1}, \ldots, \mathsf{que}_{0,m}),\\ \boldsymbol{r} &= (\mathsf{que}_{1,1}, \ldots, \mathsf{que}_{1,j-1}, \mathsf{que}_{1,i}, \mathsf{que}_{1,j+1}, \ldots, \mathsf{que}_{1,m}).\end{aligned}$$

     $\mathsf{C}$ sends $(q_\ell, r_\ell)$ to $\mathsf{S}_\ell$.
   - If $\alpha = 1$, then $\mathsf{C}$ sets

$$\begin{aligned}\boldsymbol{r} &= (\mathsf{que}_{0,1}, \ldots, \mathsf{que}_{0,j-1}, \mathsf{que}_{0,i}, \mathsf{que}_{0,j+1}, \ldots, \mathsf{que}_{0,m}),\\ \boldsymbol{q} &= (\mathsf{que}_{1,1}, \ldots, \mathsf{que}_{1,m}).\end{aligned}$$

     $\mathsf{C}$ sends $(r_\ell, q_\ell)$ to $\mathsf{S}_\ell$.

**Answer.** Upon receiving two queries $(p_0, p_1)$, each server $\mathsf{S}_\ell$ sends $\mathsf{C}$

$$\mathsf{ans}_\ell = (\mathsf{Answer}(D, p_0), \mathsf{Answer}(D, p_1)).$$

**Output.**

1. Upon receiving $\mathsf{ans}_\ell$ from each $\mathsf{S}_\ell$, $\mathsf{C}$ sets $a_\ell, b_\ell$ as

$$\mathsf{ans}_\ell = (a_\ell, b_\ell) \text{ if } \alpha = 0, \text{ and } \mathsf{ans}_\ell = (b_\ell, a_\ell) \text{ if } \alpha = 1.$$

2. $\mathsf{C}$ outputs $\boldsymbol{a} = (a_1, \ldots, a_m)$, $\boldsymbol{b} = (b_1, \ldots, b_m)$ and $\widetilde{\mathsf{st}} = (\mathsf{st}_\alpha, i, j)$.

Figure 7: A sub-protocol $\Pi_0$ in the proof of Theorem 7.8

**Notations.** Continuing the notations in Figure 7,

- Let $\Pi_0$ be a sub-protocol described in Figure 7.
- Let $\kappa \in \mathbb{N}$ be a parameter.

**Input.** A client $\mathsf{C}$ has an index $x \in [N]$ and each server $\mathsf{S}_i$ has a database $D \in \mathcal{X}^N$.

**Output.** $\mathsf{C}$ obtains a list $\mathcal{L}$.

**Protocol.**

1. $\mathsf{C}$ executes $\kappa$ independent instances of $\Pi_0$ in parallel.
2. For $u \in [\kappa]$, let $\boldsymbol{a}^{(u)} = (a_\ell^{(u)})_{\ell \in [m]}$, $\boldsymbol{b}^{(u)} = (b_\ell^{(u)})_{\ell \in [m]}$ and $\widetilde{\mathsf{st}}^{(u)} = (\mathsf{st}^{(u)}, i^{(u)}, j^{(u)})$ be the outputs of the $u$-th instance of $\Pi_0$.
3. $\mathsf{C}$ computes a graph $G$ on $m$ nodes as follows:
   (a) Let $G^{(0)}$ be the complete graph on $m$ nodes.
   (b) For each $u \in [\kappa]$, let $G^{(u)}$ be a graph obtained by removing an edge $(i^{(u)}, j^{(u)})$ from $G^{(u-1)}$ if $b_{i^{(u)}}^{(u)} \neq b_{j^{(u)}}^{(u)}$. Otherwise, let $G^{(u)} = G^{(u-1)}$.
   (c) Let $G = G^{(\kappa)}$.
4. $\mathsf{C}$ computes cliques $C_1, \ldots, C_h$ of size $k$ in $G$ as follows: Set $G_1 = G$ and $g \leftarrow 1$.
   (a) Find a clique $C_g$ of size $k$ in $G_g$. If no clique is found, break the loop.
   (b) Let $G_{g+1}$ be the graph obtained by removing the nodes in $C_g$ from $G_g$. Set $g \leftarrow g+1$ and go back to the previous step.
5. For each $g \in [h]$, $\mathsf{C}$ does the following:
   (a) $\mathsf{C}$ computes
   $$y_g^{(u)} = \mathsf{Output}(C_g, \mathsf{st}^{(u)}, (a_\ell^{(u)})_{\ell \in C_g})$$
   for all $u \in [\kappa]$.
   (b) If there exists $y \in \mathcal{X}$ such that $|\{u \in [\kappa] : y_g^{(u)} = y\}| > \kappa/2$, then $\mathsf{C}$ sets $y_g = y$. Otherwise, $\mathsf{C}$ sets $y_g$ to the default value $y_0$.
6. $\mathsf{C}$ outputs a list $\mathcal{L} = \{y_g : g \in [h]\}$.

Figure 8: An LD-PIR protocol $\Pi'$ in Theorem 7.8

On the other hand, since $i \in C_g$, there exists $j \in H$ such that there remains an edge $(i,j)$ in $G$. This implies that for any $u \in K_i$, the following event occurs:

$$\mathsf{F}_{ij}^{(u)} : \ (i^{(u)}, j^{(u)}) \neq (i,j) \vee b_i^{(u)} = \mathsf{Answer}(D, r_i^{(u)})$$

since otherwise the answer $b_i^{(u)}$ from $\mathsf{S}_i$ is compared with the answer $b_j^{(u)} = \mathsf{Answer}(D, r_i^{(u)})$ from $\mathsf{S}_j$ and then the edge $(i,j)$ is removed. We define $\mathsf{F}_{ij}$ as the event that $\mathsf{F}_{ij}^{(u)}$ occurs for all $u \in K_i$.

From the above observation, the event $\mathsf{F}$ happens only if there exist $i \in B$ and $j \in H$ such that $\mathsf{E}_i$ and $\mathsf{F}_{ij}$ occur. We have that

$$\begin{aligned}
\Pr[\mathsf{F}] &\leq \Pr[\exists i \in B, \exists j \in H : \mathsf{E}_i \wedge \mathsf{F}_{ij}] \\
&\leq \sum_{i \in B, j \in H} \Pr[\mathsf{E}_i \wedge \mathsf{F}_{ij}] \\
&\leq \sum_{i \in B, j \in H} \sum_{A : |A| \geq \frac{\kappa}{2b}} \Pr[K_i = A] \Pr[\mathsf{F}_{ij} \mid K_i = A]. \qquad (10)
\end{aligned}$$

For $i \in B$, $j \in H$ and $A$ with $|A| \geq \kappa/(2b)$, we analyze $\Pr[\mathsf{F}_{ij} \mid K_i = A]$. For simplicity, we assume that $A = \{1, 2, \ldots, v\}$, where $v = |A|$. We have that

$$\Pr[\mathsf{F}_{ij} \mid K_i = A] = \prod_{u=1}^{v} \Pr\left[\mathsf{F}_{ij}^{(u)} \;\middle|\; (K_i = A) \wedge \mathsf{F}_{ij}^{(1)} \wedge \cdots \wedge \mathsf{F}_{ij}^{(u-1)}\right]. \qquad (11)$$

Let $\pi^{(u)} = (i^{(u)}, j^{(u)}, \alpha^{(u)})$ denote randomness sampled by $\mathsf{C}$ at Step 1 of **Query** of the $u$-th instance of $\Pi_0$. Then,

$$\begin{aligned}
&\Pr\left[\mathsf{F}_{ij}^{(u)} \;\middle|\; (K_i = A) \wedge \mathsf{F}_{ij}^{(1)} \wedge \cdots \wedge \mathsf{F}_{ij}^{(u-1)}\right] \\
= &\sum_{\substack{\pi^{(1)}, \ldots, \pi^{(u-1)}, \\ \pi^{(u+1)}, \ldots, \pi^{(v)}}} \Pr\left[\pi^{(1)}, \ldots, \pi^{(u-1)}, \pi^{(u+1)}, \ldots, \pi^{(v)}\right] \\
&\times \Pr_{\pi^{(u)}}\left[\mathsf{F}_{ij}^{(u)} \;\middle|\; (K_i = A) \wedge \mathsf{F}_{ij}^{(1)} \wedge \cdots \wedge \mathsf{F}_{ij}^{(u-1)}, \pi^{(1)}, \ldots, \pi^{(u-1)}, \pi^{(u+1)}, \ldots, \pi^{(v)}\right]. \qquad (12)
\end{aligned}$$

Fix $\pi^{(1)}, \ldots, \pi^{(u-1)}, \pi^{(u+1)}, \ldots, \pi^{(v)}$. For ease of reading, let $\mathsf{COND}$ denote the condition of the probability (12). Define $\mathsf{BAD}$ be an event that $\mathsf{C}$ picks $\pi^{(u)} = (i^{(u)}, j^{(u)}, \alpha^{(u)})$ such that $j^{(u)} \neq j$. Then, we have that

$$\begin{aligned}
&\Pr_{\pi^{(u)}}\left[\mathsf{F}_{ij}^{(u)} \;\middle|\; \mathsf{COND}\right] \\
\leq \ &\Pr[\mathsf{BAD}] + \Pr\left[\overline{\mathsf{BAD}}\right] \cdot \Pr_{\pi^{(u)}}\left[\mathsf{F}_{ij}^{(u)} \;\middle|\; \mathsf{COND} \wedge \overline{\mathsf{BAD}}\right] \\
\leq \ &\frac{m-1}{m} + \frac{1}{m} \cdot \Pr_{\pi^{(u)}}\left[\mathsf{F}_{ij}^{(u)} \;\middle|\; \mathsf{COND} \wedge \overline{\mathsf{BAD}}\right]. \qquad (13)
\end{aligned}$$

We will show that

$$\Pr_{\pi^{(u)}}\left[\mathsf{F}_{ij}^{(u)} \;\middle|\; \mathsf{COND} \wedge \overline{\mathsf{BAD}}\right] \leq 1 - \frac{1}{2(m-1)}. \qquad (14)$$

Let $X$ denote the set of all $\pi^{(u)}$'s such that $\mathsf{COND} \wedge \overline{\mathsf{BAD}}$ occurs. Let $Y$ denote the subset of $X$ consisting of all $\pi^{(u)}$'s such that $\mathsf{F}_{ij}^{(u)}$ additionally occurs.

If $X = \emptyset$, then (14) clearly holds. If $X \neq \emptyset$, choose $\pi^{(u)} = (i^{(u)}, j, \alpha^{(u)}) \in X$ arbitrarily. Since $j \notin B$, when $\pi^{(u)}$ is picked, the queries seen by the adversary can be written as

$$\left\{(\mathsf{que}_{0,\ell}^{(u')}, \mathsf{que}_{1,\ell}^{(u')}) : \ell \in B, u' \in [\kappa]\right\}, \qquad (15)$$

where $\mathsf{que}_{0,\ell}^{(u')}, \mathsf{que}_{1,\ell}^{(u')}$ are queries computed at Step 2 of **Query** in the $u'$-th instance of $\Pi_0$. This implies that for any $\pi' = (i', j, \alpha')$ with $i' \neq j$ and $\alpha' \in \{0,1\}$, the queries seen by the adversary when $\pi'$ is picked are the same as (15). Since the adversary's randomness is fixed, the answers returned by corrupted servers are also the same regardless of what is picked as $(i^{(u)}, \alpha^{(u)})$ at Step 1 of **Query** of $\Pi_0$. Therefore, if $\mathsf{COND} \wedge \overline{\mathsf{BAD}}$ occurs on some $\pi^{(u)}$, then $\mathsf{COND} \wedge \overline{\mathsf{BAD}}$ occurs on every $(i', j, \alpha')$ with $i' \neq j$ and $\alpha' \in \{0,1\}$. In particular, we have $|X| = 2(m-1)$ if $X \neq \emptyset$.

We have seen that answers returned by corrupted servers are the same for any $\pi^{(u)} \in X$. We denote the answers in the $u$-th instance of $\Pi_0$ by $\{\mathsf{ans}_\ell^{(u)} = (\mathsf{ans}_{0,\ell}^{(u)}, \mathsf{ans}_{1,\ell}^{(u)}) : \ell \in B\}$. Since $u \in K_i$,

$$\mathsf{ans}_i^{(u)} = (\mathsf{ans}_{0,i}^{(u)}, \mathsf{ans}_{1,i}^{(u)}) \neq (\mathsf{Answer}(D, \mathsf{que}_{0,i}^{(u)}), \mathsf{Answer}(D, \mathsf{que}_{1,i}^{(u)})).$$

If $\mathsf{ans}_{1,i}^{(u)} \neq \mathsf{Answer}(D, \mathsf{que}_{1,i}^{(u)})$, then $(i,j,0) \notin Y$. This is because $\mathsf{F}_{ij}^{(u)}$ does not occur on $(i,j,0)$ since $(i^{(u)}, j^{(u)}) = (i,j)$ and $b_i^{(u)} = \mathsf{ans}_{1,i}^{(u)} \neq \mathsf{Answer}(D, \mathsf{que}_{1,i}^{(u)})$. Similarly, if $\mathsf{ans}_{0,i}^{(u)} \neq \mathsf{Answer}(D, \mathsf{que}_{0,i}^{(u)})$, then $(i,j,1) \notin Y$. Therefore, if $X \neq \emptyset$,

$$\Pr_{\pi^{(u)}}\left[\mathsf{F}_{ij}^{(u)} \;\middle|\; \mathsf{COND} \wedge \overline{\mathsf{BAD}}\right] = \frac{|Y|}{|X|} \leq \frac{2(m-1)-1}{2(m-1)},$$

which implies (14). From (13), we have that

$$\Pr_{\pi^{(u)}}\left[\mathsf{F}_{ij}^{(u)} \;\middle|\; \mathsf{COND}\right] \leq 1 - \frac{1}{2m(m-1)}.$$

Combining (10), (11), and (12), we conclude that the proability of $\mathsf{F}$ is upper bounded by $|H||B|(1 - 1/(2m(m-1)))^{\kappa/(2b)} \leq b(m-b)\exp(-\kappa/(4m^2b)) \leq \exp(-\Omega((\ln\lambda)^2))$, which is negligible in $\lambda$.

**Complexity.** The communication complexity of $\Pi'$ is $\kappa$ times larger than that of $\Pi_0$, which is at most twice larger than that of $\Pi$. Hence, we have $\mathrm{Comm}(\Pi') = \mathrm{Comm}(\Pi) \cdot 2\kappa = \mathrm{Comm}(\Pi) \cdot \widetilde{O}(m^2 b)$. All the steps except for Step 5 of $\Pi'$ can be done in time $O(\kappa) \cdot \mathrm{Comp}(\Pi) = \widetilde{O}(m^2 b) \cdot \mathrm{Comp}(\Pi)$. Since $h \leq L$, the computation complexity of Step 5 of $\Pi'$ is at most $L \cdot \tau_{m,m-b}$. Thus, we have $\mathrm{Comp}(\Pi') = \widetilde{O}(m^2 b) \cdot \mathrm{Comp}(\Pi) + L \cdot \tau_{m,m-b} = \widetilde{O}(m^2 b) \cdot \mathrm{Comp}(\Pi) + O(\binom{m}{b} m^2 L)$.

# F Instantiations

## F.1 LD-PIR from Efremenko's PIR

The protocol of Efremenko [20] also achieves sub-polynomial communication complexity in the database size $N$. Compared to Dvir-Gopi's protocol [19], it has a better dependency of communication complexity on a privacy threshold $t$ at the cost of a larger number of servers. Again, the original protocol in [20] assumes $t = 1$ but can be extended to a $t$-private protocol for $t \geq 2$ by using the privacy amplification technique in [3].

**Proposition F.1.** *There exists a 1-round $t$-private $3^t$-server PIR protocol $\mathsf{Efr}_t$ such that $\mathrm{Comm}(\mathsf{Efr}_t) = h_N 2^{O(t)} \log|\mathcal{X}|$, where $h_N = \exp(O(\sqrt{\log N \log\log N})) = N^{o(1)}$ is the same value introduced in Proposition 8.1.*

We denote by $\mathsf{X}\text{-}\mathsf{Efr}_t$ an LD-PIR protocol obtained by applying a generic construction

$$\mathsf{X} \in \{\mathsf{Exh}, \mathsf{Local1}, \mathsf{Local2}, \mathsf{Clique1}, \mathsf{Clique2}\}$$

to $\mathsf{Efr}_t$. We use the fact that $k'/\ln k' \geq 3^t$ if $k' = 2t3^t$ to apply $\mathsf{X} \in \{\mathsf{Local2}, \mathsf{Clique2}\}$. We summarize the performance of these protocols in Table 3.

Among all of them, $\mathsf{Exh}\text{-}\mathsf{Efr}_t$, $\mathsf{Local1}\text{-}\mathsf{Efr}_t$, and $\mathsf{Clique1}\text{-}\mathsf{Efr}_t$ achieve the minimum number of servers $m \geq m_L(b, 3^t)$. Notably, they even achieve the minimum number of servers required for $(b, L)$-list-decoding, i.e., $m > b(L+1)/L$, for a sufficiently large number of errors $b > L3^t$.

Table 3: Comparison of one-round LD-PIR obtained from Efremenko's PIR

| | | Overhead | | |
|---|---|---|---|---|
| Protocol | Servers | Communication | Computation | Remark |
| $\mathsf{Exh\text{-}Efr}_t$ | $m_L(b, 3^t)$ | $\binom{m}{3^t}3^{3t}$ | $\binom{m}{b}\binom{m-b}{3^t}3^{3t}$ | |
| $\mathsf{Local1\text{-}Efr}_t$ | $m_L(b, 3^t)$ | $m^2b3^te^{3^t}$ | $m^2b3^te^{3^t}$ | 1-local error model |
| $\mathsf{Local2\text{-}Efr}_t$ | $m_L(b, 2t3^t)$ | $m^2bt3^t$ | $m^2bt3^t$ | 1-local error model |
| $\mathsf{Clique1\text{-}Efr}_t$ | $m_L(b, 3^t)$ | $m^3b3^te^{3^t}$ | $\binom{m}{b}m^3b3^te^{3^t}L$ | |
| $\mathsf{Clique2\text{-}Efr}_t$ | $m_L(b, 2t3^t)$ | $m^3bt3^t$ | $\binom{m}{b}m^3bt3^tL$ | |

$m$ is the number of servers, $b$ is the number of errors, $L$ is the list size, and $t$ is a privacy threshold. "Servers" refers the minimum number of servers required and $m_L(b, k)$ denotes the smallest integer $n$ such that $n > b(L+1)/L$ and $n \geq b + k$. "Overhead" refers to the ratio of the communication (resp. computation) complexity relative to that of $\mathsf{Efr}_t$, omitting polylogarithmic factors in $m$ and $\lambda$.

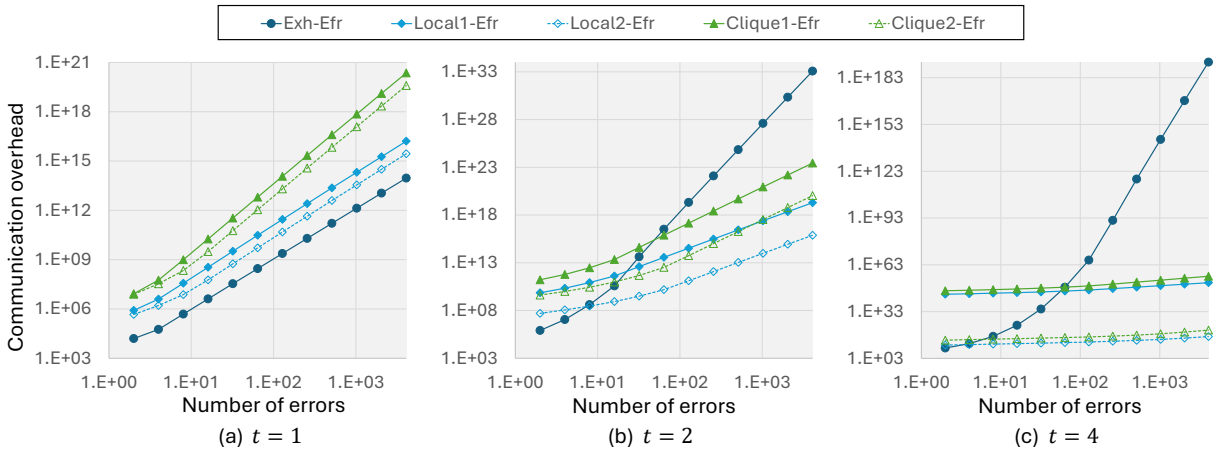


Figure 9: Communication overhead of one-round LD-PIR obtained from Efremenko's PIR in the setting of $L = 2$ and $t \in \{1, 2, 4\}$

$\mathsf{Clique2\text{-}Efr}_t$ achieves the asymptotically lowest communication and computation complexity in the non-local error model and $\mathsf{Local2\text{-}Efr}_t$ is the lowest in the local error model. More concretely, Figure 9 shows the ratio of the communication complexity of each LD-PIR protocol $\mathsf{X\text{-}Efr}_t$ ($\mathsf{X} \in \{\mathsf{Exh}, \mathsf{Local1}, \mathsf{Local2}, \mathsf{Clique1}, \mathsf{Clique2}\}$), relative to that of the underlying regular PIR protocol $\mathsf{Efr}_t$. We fix $L = 2$ and the failure probability $\epsilon = 2^{-60}$. Generally speaking, we observe a trend similar to that of the instantiations from $\mathsf{DG}_t$ described in Figure 2. Due to a larger factor of $\binom{m}{3^t} = O(m^{3^t})$, the communication complexity of $\mathsf{Exh\text{-}Efr}_t$ becomes prohibitive more quickly than $\mathsf{Exh\text{-}DG}_t$.

## F.2 LD-PIR from Woodruff-Yekhanin's PIR

The protocol of [42] has higher communication complexity $O(N^{1/d})$ for a parameter $d$ but the number of servers scales linearly in $t$. The protocol is known to natively satisfy the robustness property.

**Proposition F.2.** *For any $d, k \in \mathbb{N}$ such that*

$$m \geq k \geq \frac{dt+1}{2},$$

*there exists a 1-round $t$-private $k$-robust $m$-server PIR protocol $\mathsf{WY}_{t,d}$ such that $\mathrm{Comm}(\mathsf{WY}_{t,d}) = \widetilde{O}(N^{1/d}md\log|\mathcal{X}|)$.*

Table 4: Comparison of one-round LD-PIR obtained from Woodruff-Yekhanin's PIR

| | | Overhead | | |
|---|---|---|---|---|
| Protocol | Servers | Communication | Computation | Remark |
| Exh-WY$_{t,d}$ | $m_L(b,k)$ | $\binom{m}{k}k^3$ | $\binom{m}{b}\binom{m-b}{k}k^3$ | |
| Local-WY$_{t,d}$ | $m_L(b,k)$ | $mb$ | $mb$ | 1-local error model |
| Clique-WY$_{t,d}$ | $m_L(b,k)$ | $m^2b$ | $\binom{m}{b}m^2bL$ | |

$m$ is the number of servers, $b$ is the number of errors, $L$ is the list size, $t$ is a privacy threshold, and $k = \lceil (dt+1)/2 \rceil$. "Servers" refers the minimum number of servers required and $m_L(b,k)$ denotes the smallest integer $n$ such that $n > b(L+1)/L$ and $n \geq b+k$. "Overhead" refers to the ratio of the communication (resp. computation) complexity relative to that of WY$_{t,d}$, omitting polylogarithmic factors in $m$ and $\lambda$.

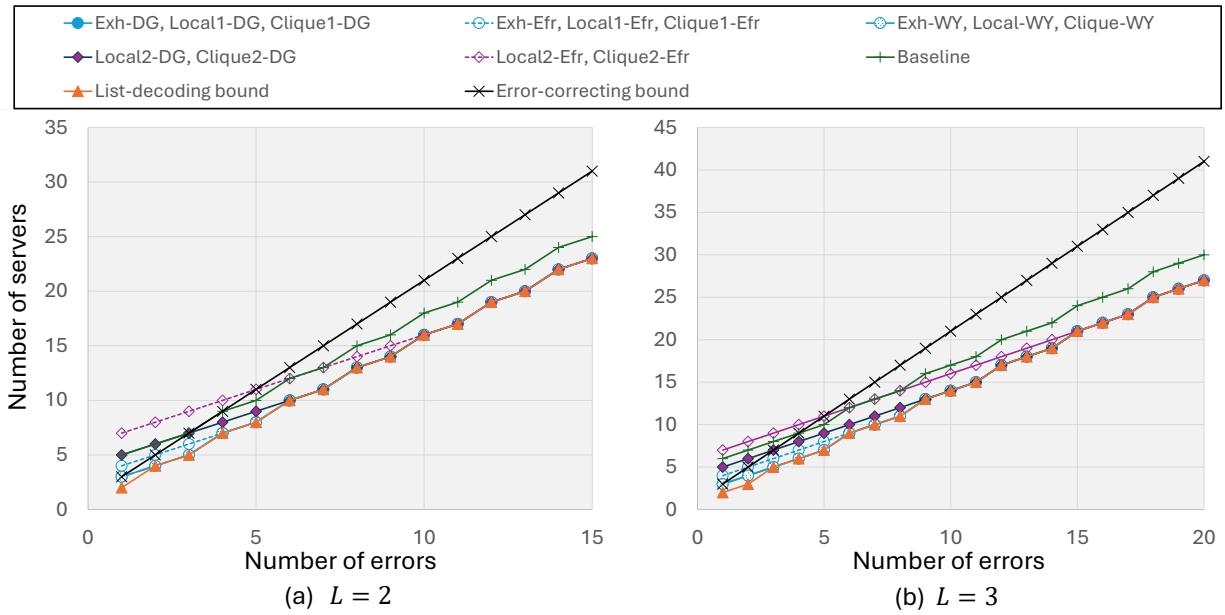


Figure 10: The number of servers in one-round LD-PIR in the setting of $L \in \{2,3\}$ and $t = 1$

We denote by Exh-WY$_{t,d}$ an LD-PIR protocol obtained by applying a generic construction Exh. Since robustness comes for free in WY$_{t,d}$, we do not need to use the regular-to-robust compilers of [6] to apply Theorems 7.5 and 7.8. We thus denote by Local-WY$_{t,d}$ and Clique-WY$_{t,d}$ protocols obtained from Theorems 7.5 and 7.8, respectively. We summarize the performance of these protocols in Table 4.

All of them achieve the same number of servers $m \geq m_L(b,k)$, where $k = \lceil (dt+1)/2 \rceil$. Notably, they even achieve the minimum number of servers required for $(b,L)$-list-decoding, i.e., $m > b(L+1)/L$, for a sufficiently large number of errors $b > L(dt+1)/2$.

Clique-WY$_{t,d}$ achieves the asymptotically lowest communication and computation complexity in the non-local error model and Local-WY$_{t,d}$ is the lowest in the local error model.

Finally, Figure 10 shows the changes in the minimum number $m$ of servers required by each LD-PIR protocol with respect to the number $b$ of errors, where we fix $t = 1$, $d = 3$ and set $L \in \{2,3\}$. Among all of them, X-DG$_t$ (X $\in$ {Exh, Local1, Clique1}) and X-WY$_{t,d}$ (X $\in$ {Exh, Local, Clique}) are the first whose number of servers falls below the error-correcting bound $2b+1$ and approaches the list decoding bound $b(L+1)/L = 2b/3$. Then, the number of servers in X-Efr$_t$ (X $\in$ {Exh, Local1, Clique1}), X-DG$_t$ (X $\in$ {Local2, Clique2}), Baseline$_{t,d}$, and X-Efr$_t$ (X $\in$ {Local2, Clique2}) become smaller than the error-correcting bound in this order. We note that the number of servers in Baseline$_{t,d}$ never converges to the list decoding bound.

Table 5: The minimum number $b$ of errors such that the number $m$ of servers required becomes smaller than the error-correcting bound $2b+1$. The numbers in bold indicate the minimum value for each parameter setting. The list size is fixed to $L = 2$.

| $(t, d)$ | $(1,3)$ | $(2,3)$ | $(4,3)$ | $(1,9)$ | $(2,9)$ | $(4,9)$ |
|---|---|---|---|---|---|---|
| $\mathsf{Baseline}_{t,d}$ | 5 | 10 | 19 | 14 | 28 | 55 |
| $\mathsf{Exh\text{-}DG}_t$, $\mathsf{Local1\text{-}DG}_t$, $\mathsf{Clique1\text{-}DG}_t$ | **2** | **4** | 16 | **2** | **4** | **16** |
| $\mathsf{Exh\text{-}Efr}_t$, $\mathsf{Local1\text{-}Efr}_t$, $\mathsf{Clique1\text{-}Efr}_t$ | 3 | 9 | 81 | 3 | 9 | 81 |
| $\mathsf{Exh\text{-}WY}_{t,d}$, $\mathsf{Local\text{-}WY}_{t,d}$, $\mathsf{Clique\text{-}WY}_{t,d}$ | **2** | **4** | **7** | 5 | 10 | 19 |
| $\mathsf{Local2\text{-}DG}_t$, $\mathsf{Clique2\text{-}DG}_t$ | 4 | 16 | 128 | 4 | 16 | 128 |
| $\mathsf{Local2\text{-}Efr}_t$, $\mathsf{Clique2\text{-}Efr}_t$ | 6 | 36 | 648 | 6 | 36 | 648 |

More concretely, Table 5 shows the minimum number $b$ of errors such that the number $m$ of servers required by each protocol becomes smaller than the error-correcting bound. That is, it shows the minimum value of $b$ such that $m_L(b,k) < 2b+1$, where $k$ is specified by the protocol. $\mathsf{X\text{-}DG}_t$ ($\mathsf{X} \in \{\mathsf{Exh}, \mathsf{Local1}, \mathsf{Clique1}\}$) fall below the error-correcting bound the most quickly, regardless of the choice of $d$.

# G Details on Unique Decoding with a Trusted Setup

## G.1 Formal Definitions

First, we present the formal definition of PIR *with a trusted setup*. In this section, we consider computational security and assume probabilistic polynomial-time (PPT) adversaries.

**Definition G.1.** *An $\ell$-round $m$-server PIR protocol with a trusted setup is a tuple of four polynomial-time algorithms $\Pi = (\mathsf{Setup}, \mathsf{Query}, \mathsf{Answer}, \mathsf{Output})$, where:*

- $\mathsf{Setup}(1^\lambda, D) \to (\mathsf{pp}, \widetilde{D}, \mathsf{dig}, \mathsf{aux})$*:* $\mathsf{Setup}$ *is a randomized algorithm that takes a security parameter $\lambda$ and a database $D \in \mathcal{X}^N$ as input, and outputs public parameters $\mathsf{pp}$, an augmented database $\widetilde{D}$, a digest $\mathsf{dig}$, and auxiliary information $\mathsf{aux}$. We assume that $\mathsf{pp}$ is a common input to all the other algorithms. For notational simplicity, we omit $\mathsf{pp}$ from the inputs of the other algorithms.*
- $\mathsf{Query}(x, \mathsf{st}^{(j-1)}, (\mathsf{ans}_i^{(j-1)})_{i\in[m]}) \to ((\mathsf{que}_i^{(j)})_{i\in[m]}, \mathsf{st}^{(j)})$*:* $\mathsf{Query}$ *is a randomized algorithm that takes an index $x \in [N]$, a state $\mathsf{st}^{(j-1)}$ and answers $(\mathsf{ans}_i^{(j-1)})_{i\in[m]}$ in round $j-1$ as input, and outputs queries $(\mathsf{que}_i^{(j)})_{i\in[m]}$ and a state $\mathsf{st}^{(j)}$ in round $j$, where we define $\mathsf{st}^{(0)}$, $\mathsf{ans}_i^{(0)}$ as the empty string;*
- $\mathsf{Answer}(\widetilde{D}, \mathsf{que}_i^{(j)}) \to \mathsf{ans}_i^{(j)}$*:* $\mathsf{Answer}$ *is a deterministic algorithm that takes an augmented database $\widetilde{D}$ and a query $\mathsf{que}_i^{(j)}$ in round $j$ as input, and outputs an answer $\mathsf{ans}_i^{(j)}$ in round $j$;*
- $\mathsf{Output}(\mathsf{st}^{(\ell)}, (\mathsf{ans}_i^{(\ell)})_{i\in[m]}, \mathsf{dig}) \to y$*:* $\mathsf{Output}$ *is a deterministic algorithm that takes a state $\mathsf{st}^{(\ell)}$, answers $(\mathsf{ans}_i^{(\ell)})_{i\in[m]}$ in round $\ell$ and a digest $\mathsf{dig}$ as input, and outputs $y \in \mathcal{X}$;*

*such that $\Pi$ satisfies the following property:*

**Correctness.** *For any $\lambda \in \mathbb{N}$, any $D = (D_1, \ldots, D_N) \in \mathcal{X}^N$ and any $x \in [N]$, it holds that*

$$\Pr\left[\mathsf{Output}(\mathsf{st}^{(\ell)}, (\mathsf{ans}_i^{(\ell)})_{i\in[m]}), \mathsf{dig}) = D_x\right] = 1,$$

*where*

$$\begin{aligned}(\mathsf{pp}, \widetilde{D}, \mathsf{dig}, \mathsf{aux}) &\leftarrow \mathsf{Setup}(1^\lambda, D),\\ ((\mathsf{que}_i^{(j)})_{i\in[m]}, \mathsf{st}^{(j)}) &\leftarrow \mathsf{Query}(x, \mathsf{st}^{(j-1)}, (\mathsf{ans}_i^{(j-1)})_{i\in[m]}),\\ \mathsf{ans}_i^{(j)} &\leftarrow \mathsf{Answer}(\widetilde{D}, \mathsf{que}_i^{(j)})\end{aligned}$$

*for all* $j \in [\ell]$ *and* $i \in [m]$.

*Define* the setup cost *of* $\Pi$ *as the bit-length* $|\mathsf{dig}|$ *of a digest* $\mathsf{dig}$.

We extend the privacy of PIR to the setting of PIR with a trusted setup.

**Definition G.2.** *Let* $\Pi = (\mathsf{Setup}, \mathsf{Query}, \mathsf{Answer}, \mathsf{Output})$ *be an* $\ell$*-round* $m$*-server PIR protocol with trusted setup. We say that* $\Pi$ *is* $t$-private *if it satisfies the following requirements:*

**Privacy.** *For any stateful PPT algorithm* $\mathcal{A}$*, the distributions of* $\mathsf{Priv}^0_{\Pi,\mathcal{A}}(\lambda)$ *and* $\mathsf{Priv}^1_{\Pi,\mathcal{A}}(\lambda)$ *are computationally indistinguishable, where for* $\alpha \in \{0,1\}$*,* $\mathsf{Priv}^\alpha_{\Pi,\mathcal{A}}(\lambda)$ *is the output* $\alpha'$ *of* $\mathcal{A}$ *in the following experiment* $\mathrm{EXPT}^{\text{Priv-s},\alpha}_{\Pi,\mathcal{A}}$*:*

1. $\mathcal{A}(1^\lambda)$ *outputs* $(D,T)$ *such that* $D \in \mathcal{X}^N$ *and* $T \subseteq [m]$ *is of size at most* $t$.
2. *Let* $(\mathsf{pp}, \widetilde{D}, \mathsf{dig}, \mathsf{aux}) \leftarrow \mathsf{Setup}(1^\lambda, D)$ *and give* $\mathsf{pp}, \widetilde{D}$ *to* $\mathcal{A}$.
3. $\mathcal{A}$ *outputs* $(x_0, x_1)$*, where* $x_0, x_1 \in [N]$.
4. *For each* $j = 1, 2, \ldots, \ell$,
   (a) *Let* $((\mathsf{que}_i^{(j)})_{i\in[m]}, \mathsf{st}^{(j)}) \leftarrow \mathsf{Query}(x_\alpha, \mathsf{st}^{(j-1)}, (\mathsf{ans}_i^{(j-1)})_{i\in[m]})$ *and give* $(\mathsf{que}_i^{(j)})_{i\in T}$ *to* $\mathcal{A}$.
   (b) *If* $j < \ell$*,* $\mathcal{A}$ *outputs* $(\mathsf{ans}_i^{(j)})_{i\in T}$*. Let* $\mathsf{ans}_i^{(j)} \leftarrow \mathsf{Answer}(\widetilde{D}, \mathsf{que}_i^{(j)})$ *for* $i \notin T$.
   (c) *If* $j = \ell$*,* $\mathcal{A}$ *outputs a bit* $\alpha' \in \{0,1\}$.

We extend the notion of LD-PIR to the setting of PIR with a trusted setup.

**Definition G.3.** *Let* $\Pi = (\mathsf{Setup}, \mathsf{Query}, \mathsf{Answer}, \mathsf{Output})$ *be an* $\ell$*-round* $m$*-server PIR protocol with a trusted setup. We say that* $\Pi$ *is* $(b, L)$-list-decodable *if* $\mathsf{Output}$ *is allowed to output a subset* $\mathcal{L} \subseteq \mathcal{X}$ *and* $\Pi$ *satisfies the following requirements:*

**List decoding.** *There exists a negligible function* $\epsilon = \epsilon(\lambda)$ *such that for any* $\lambda \in \mathbb{N}$ *and any stateful PPT algorithm* $\mathcal{A}$*, it holds that* $\Pr[\mathsf{LD}_{\Pi,\mathcal{A}}(\lambda) = 1] \le \epsilon$*, where* $\mathsf{LD}_{\Pi,\mathcal{A}}(\lambda)$ *is the output of the following experiment* $\mathrm{EXPT}^{\text{LD-s}}_{\Pi,\mathcal{A}}(\lambda)$*:*

1. $\mathcal{A}(1^\lambda)$ *outputs* $(D,T)$ *such that* $D \in \mathcal{X}^N$ *and* $T \subseteq [m]$ *is of size at most* $b$.
2. *Let* $(\mathsf{pp}, \widetilde{D}, \mathsf{dig}, \mathsf{aux}) \leftarrow \mathsf{Setup}(1^\lambda, D)$ *and give* $\mathsf{pp}, \widetilde{D}$ *to* $\mathcal{A}$.
3. $\mathcal{A}$ *outputs* $x \in [N]$.
4. *For each* $j = 1, 2, \ldots, \ell$,
   (a) *Let* $((\mathsf{que}_i^{(j)})_{i\in[m]}, \mathsf{st}^{(j)}) \leftarrow \mathsf{Query}(x, \mathsf{st}^{(j-1)}, (\mathsf{ans}_i^{(j-1)})_{i\in[m]})$ *and give* $(\mathsf{que}_i^{(j)})_{i\in T}$ *to* $\mathcal{A}$.
   (b) $\mathcal{A}$ *outputs* $(\mathsf{ans}_i^{(j)})_{i\in T}$*. Let* $\mathsf{ans}_i^{(j)} \leftarrow \mathsf{Answer}(\widetilde{D}, \mathsf{que}_i^{(j)})$ *for* $i \notin T$.
5. *Let* $\mathcal{L} \leftarrow \mathsf{Output}(\mathsf{st}^{(\ell)}, (\mathsf{ans}_i^{(\ell)})_{i\in[m]}, \mathsf{dig})$*. Return* $0$ *if* $D_x \in \mathcal{L}$ *and* $|\mathcal{L}| \le L$*, and otherwise return* $1$.

## G.2 Details on the Construction Based on Digital Signature

We recall the notion of digital signature.

**Definition G.4** (Digital signature)**.** *A digital signature scheme is a tuple of three polynomial-time algorithms* $\Sigma = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify})$*, where:*

- $\mathsf{Gen}(1^\lambda) \to (\mathsf{pk}, \mathsf{sk})$*:* $\mathsf{Gen}$ *is an algorithm that takes a security parameter* $\lambda$ *as input and outputs a public key* $\mathsf{pk}$ *and a secret key* $\mathsf{sk}$*;*
- $\mathsf{Sign}(\mathsf{sk}, x) \to \sigma$*:* $\mathsf{Sign}$ *is an algorithm that takes a secret key* $\mathsf{sk}$ *and a message* $x$ *as input, and outputs a signature* $\sigma$*;*
- $\mathsf{Verify}(\mathsf{pk}, x, \sigma) \to b$*:* $\mathsf{Verify}$ *is an algorithm that takes a public key* $\mathsf{pk}$*, a message* $x$ *and a signature* $\sigma$ *as input, and outputs a bit* $b$*;*

**Notations.**

- Let $\Sigma = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify})$ be a digital signature scheme.
- Let $\Pi = (\mathsf{Query}, \mathsf{Answer}, \mathsf{Output})$ be a $t$-private $\ell$-round $m$-server $(b, L)$-LD-PIR protocol.

$\mathsf{Setup}'(1^\lambda, D)$**.** Given a security parameter $\lambda$ and a database $D \in \mathcal{X}^N$:

1. $(\mathsf{pk}, \mathsf{sk}) \leftarrow \Sigma.\mathsf{Gen}(1^\lambda)$.
2. For each $x \in [N]$, let $\sigma_x \leftarrow \Sigma.\mathsf{Sign}(\mathsf{sk}, (D_x, x))$ be a signature of a message $(D_x, x)$.
3. Output an augmented database $\widetilde{D} = (\widetilde{D}_1, \ldots, \widetilde{D}_N)$ where $\widetilde{D}_x = (D_x, \sigma_x)$, and a digest $\mathsf{dig} = \mathsf{pk}$.

$\mathsf{Query}'(x, \mathsf{st}^{(j-1)}, (\mathsf{ans}_i^{(j-1)})_{i\in[m]})$**.** The same as $\mathsf{Query}(x, \mathsf{st}^{(j-1)}, (\mathsf{ans}_i^{(j-1)})_{i\in[m]})$.

$\mathsf{Answer}'(\widetilde{D}, \mathsf{que}_i^{(j)})$**.** The same as $\mathsf{Answer}'(D, \mathsf{que}_i^{(j)})$ except that an answer is computed on the augmented one $\widetilde{D}$ instead of $D$.

$\mathsf{Output}'(\mathsf{st}^{(\ell)}, (\mathsf{ans}_i^{(\ell)})_{i\in[m]}, \mathsf{dig})$**.** Given a state $\mathsf{st}^{(\ell)}$, answers $\mathsf{ans}_i^{(\ell)}$ and a digest $\mathsf{dig} = \mathsf{pk}$:

1. Compute a list
$$\mathcal{L}_0 \leftarrow \mathsf{Output}(\mathsf{st}^{(\ell)}, (\mathsf{ans}_i^{(\ell)})_{i\in[m]}).$$
2. Let $\mathcal{L}_1$ be the list of all valid pairs of messages and signatures in $\mathcal{L}_0$, that is,
$$\mathcal{L}_1 = \{(y, \sigma) \in \mathcal{L}_0 : \Sigma.\mathsf{Verify}(\mathsf{pk}, (y, x), \sigma) = 1\}.$$
3. Output $\mathcal{L} = \{y : (y, \sigma) \in \mathcal{L}_1\}$.

Figure 11: A construction of $\Pi'$ based on a digital signature scheme

*satisfying the following properties:*

**Completeness.** *For any $\lambda \in \mathbb{N}$ and any message $x$, it holds with probability 1 that $\mathsf{Verify}(\mathsf{pk}, x, \sigma) = 1$, where $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda)$ and $\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}, x)$;*

**Unforgeability.** *There exists a negligible function $\mathsf{negl}(\lambda)$ such that for any stateful algorithm $\mathcal{A}$ and any $\lambda \in \mathbb{N}$, it holds that $\Pr[\mathsf{UF}_{\Sigma,\mathcal{A}}(\lambda) = 1] \leq \mathsf{negl}(\lambda)$, where $\mathsf{UF}_{\Sigma,\mathcal{A}}(\lambda)$ is the output of the following experiment:*

1. *Let $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda)$ and give $\mathsf{pk}$ to $\mathcal{A}$.*
2. *$\mathcal{A}$ picks a polynomial number of messages $\{x_i\}_i$.*
3. *For each $i$, let $\sigma_i \leftarrow \mathsf{Sign}(\mathsf{sk}, x_i)$ and give $\sigma_i$ to $\mathcal{A}$.*
4. *$\mathcal{A}$ outputs $(x, \sigma)$. Return 1 if $x \neq x_i$ for all $i$ and $\mathsf{Verify}(\mathsf{pk}, x, \sigma) = 1$, and otherwise return 0.*

We show our construction of $(b, 1)$-LD-PIR with a trusted setup based on digital signature.

**Theorem G.5.** *Let $\Pi$ be a $t$-private $\ell$-round $m$-server $(b, L)$-LD-PIR protocol. Let $\Sigma$ be a digital signature scheme. Then, there exists a $t$-private $\ell$-round $m$-server $(b, 1)$-LD-PIR protocol $\Pi'$ with a trusted setup such that $\mathrm{Comm}(\Pi') = |\sigma| \cdot \mathrm{Comm}(\Pi)$ and the setup cost is $|\mathsf{pk}|$, where $|\sigma|$ and $|\mathsf{pk}|$ are the bit-lengths of a signature and a public key of $\Sigma$, respectively.*

*Proof.* We describe the construction of $\Pi'$ in Figure 11.

**Complexity.** By construction, the communication complexity of $\Pi'$ is $|\sigma|$ times larger than $\Pi$ since servers' answers are computed on $\widetilde{D}$ instead of $D$. The setup cost is the bit-length of a public key $\mathsf{pk}$ of $\Sigma$ since the additional information received by a client is $\mathsf{pk}$ only.

**Privacy.** We reduce the privacy of $\Pi'$ to that of $\Pi$. Let $\mathcal{A}'$ be an adversary against the privacy of $\Pi'$. We construct an adversary $\mathcal{A}$ against the privacy of $\Pi$ as follows.

$\mathcal{A}$

**Setup.** On input $1^\lambda$,

1. Run $(D, T) \leftarrow \mathcal{A}'(1^\lambda)$, where $D \in \mathcal{X}^N$ and $T$ is a subset of $[m]$ of size $t$.
2. Run $(\mathsf{pp}, \widetilde{D}, \mathsf{dig}, \mathsf{aux}) \leftarrow \mathsf{Setup}'(1^\lambda, D)$.
3. Give $\mathsf{pp}, \widetilde{D}$ to $\mathcal{A}'$.
4. Run $(x_0, x_1) \leftarrow \mathcal{A}'$.
5. Output $(x_0, x_1, \widetilde{D}, T)$.

**Protocol.** During the execution of $\Pi$, $\mathcal{A}$ emulates the behavior of $\mathcal{A}'$. More precisely, to compute an answer of $\mathsf{S}_i$ such that $i \in T$, $\mathcal{A}$ relays the answer of $\mathsf{S}_i$ computed by $\mathcal{A}'$.

**Output.** To output a guess bit $\alpha'$, $\mathcal{A}$ relays the bit output by $\mathcal{A}'$.

It can be seen that $\mathsf{Priv}^{\alpha}_{\Pi,\mathcal{A}}(\lambda) = \mathsf{Priv}^{\alpha}_{\Pi',\mathcal{A}'}(\lambda)$ for $\alpha \in \{0, 1\}$, since the information that $\mathcal{A}$ gives to $\mathcal{A}'$ is exactly the same as the one that $\mathcal{A}'$ sees during the experiment $\mathrm{EXPT}^{\text{Priv-s},\alpha}_{\Pi',\mathcal{A}'}(\lambda)$. Therefore, the distributions of $\mathsf{Priv}^{0}_{\Pi',\mathcal{A}'}(\lambda)$ and $\mathsf{Priv}^{1}_{\Pi',\mathcal{A}'}(\lambda)$ are statistically (resp. computationally) indistinguishable if $\mathsf{Priv}^{0}_{\Pi,\mathcal{A}}(\lambda)$ and $\mathsf{Priv}^{1}_{\Pi,\mathcal{A}}(\lambda)$ are statistically (resp. computationally) indistinguishable.

**Unique decoding.** We reduce the $(b, 1)$-list decoding property of $\Pi'$ to the $(b, L)$-list decoding property of $\Pi$ and the unforgeability of $\Sigma$. Let $\mathcal{A}'$ be an adversary against the $(b, 1)$-list decoding property of $\Pi'$.

Define $\mathsf{BAD}_1$ as an event that $D_x \notin \mathcal{L}$ during the experiment $\mathrm{EXPT}^{\text{LD-s}}_{\Pi',\mathcal{A}'}(\lambda)$. Also, define $\mathsf{BAD}_2$ as an event that $|\mathcal{L}| > 1$ during the experiment $\mathrm{EXPT}^{\text{LD-s}}_{\Pi',\mathcal{A}'}(\lambda)$. Since $\Pr[\mathsf{LD}_{\Pi',\mathcal{A}'}(\lambda) = 1] = \Pr[\mathsf{BAD}_1 \vee \mathsf{BAD}_2] \leq \Pr[\mathsf{BAD}_1] + \Pr\left[\overline{\mathsf{BAD}_1} \wedge \mathsf{BAD}_2\right]$, it is sufficient to show that

$$\Pr[\mathsf{BAD}_1] \leq \mathsf{negl}(\lambda)\,, \tag{16}$$

$$\Pr\left[\overline{\mathsf{BAD}_1} \wedge \mathsf{BAD}_2\right] \leq \mathsf{negl}(\lambda) \tag{17}$$

for a negligible function $\mathsf{negl}(\lambda)$.

To prove (16), we construct an adversary $\mathcal{A}$ against the $(b, L)$-list decoding property of $\Pi$ as follows.

$\mathcal{A}$

**Setup.** On input $1^\lambda$,

1. Run $(D, T) \leftarrow \mathcal{A}'(1^\lambda)$, where $D \in \mathcal{X}^N$ and $T$ is a subset of $[m]$ of size $b$.
2. Run $(\mathsf{pp}, \widetilde{D}, \mathsf{dig}, \mathsf{aux}) \leftarrow \mathsf{Setup}'(1^\lambda, D)$.
3. Give $\mathsf{pp}, \widetilde{D}$ to $\mathcal{A}'$.
4. Run $x \leftarrow \mathcal{A}'$.
5. Output $(x, \widetilde{D}, T)$.

**Protocol.** During the execution of $\Pi$, $\mathcal{A}$ emulates the behavior of $\mathcal{A}'$. More precisely, to compute an answer of $\mathsf{S}_i$ such that $i \in T$, $\mathcal{A}$ relays the answer of $\mathsf{S}_i$ computed by $\mathcal{A}'$.

Consider the experiment $\mathrm{EXPT}^{\mathrm{LD}}_{\Pi,\mathcal{A}}(\lambda)$ defined in Definition A.2. Let $\mathcal{L}_0$ be a list output by a client in $\mathrm{EXPT}^{\mathrm{LD}}_{\Pi,\mathcal{A}}(\lambda)$. It follows from the constructions of $\Pi$ and $\mathcal{A}$ that the list $\mathcal{L}_0$ computed in $\mathrm{EXPT}^{\mathrm{LD}}_{\Pi,\mathcal{A}}(\lambda)$ is the

same as $\mathcal{L}_0$ obtained at Step 1 of $\mathsf{Output}'$ in $\mathrm{EXPT}^{\text{LD-}s}_{\Pi',\mathcal{A}'}(\lambda)$. If $\mathsf{BAD}_1$ happens, then $(D_x, \sigma_x)$ is not included in $\mathcal{L}_0$ in $\mathrm{EXPT}^{\text{LD-}s}_{\Pi',\mathcal{A}'}(\lambda)$, which implies that $\mathsf{LD}_{\Pi,\mathcal{A}}(\lambda) = 1$. Since $\Pi$ is $(b, L)$-list-decodable, we have that $\Pr[\mathsf{BAD}_1]$ is negligible.

To prove (17), we construct an adversary $\mathcal{A}_\Sigma$ against the unforgeability of $\Sigma$ as follows.

$\mathcal{A}_\Sigma$

1. On input $1^\lambda$, run $(D, T) \leftarrow \mathcal{A}'(1^\lambda)$, where $D \in \mathcal{X}^N$ and $T$ is a subset of $[m]$ of size $b$.
2. Simulate the output of $\mathsf{Setup}'(1^\lambda, D)$ by querying the signing oracle. That is, upon receiving a public key $\mathsf{pk}$ of $\Sigma$, set $\mathsf{dig} = \mathsf{pk}$. For each $x \in [N]$, query a signature $\sigma_x$ of a message $(D_x, x)$ and set $\widetilde{D} = (D_x, x)_{x\in[N]}$. Then, give $\widetilde{D}$ to $\mathcal{A}'$.
3. Run $x \leftarrow \mathcal{A}'$.
4. Locally simulate the execution of $\Pi'$ on a client input $x$. That is, for each $j = 1, 2, \ldots, \ell$,
   (a) Let $((\mathsf{que}_i^{(j)})_{i\in[m]}, \mathsf{st}^{(j)}) \leftarrow \mathsf{Query}'(x, \mathsf{st}^{(j-1)}, (\mathsf{ans}_i^{(j-1)})_{i\in[m]})$ and give $(\mathsf{que}_i^{(j)})_{i\in T}$ to $\mathcal{A}'$.
   (b) $\mathcal{A}'$ outputs $(\mathsf{ans}_i^{(j)})_{i\in T}$. Let $\mathsf{ans}_i^{(j)} \leftarrow \mathsf{Answer}'(\widetilde{D}, \mathsf{que}_i^{(j)})$ for $i \notin T$.
5. Based on $\mathsf{st}^{(\ell)}$, $(\mathsf{ans}_i^{(\ell)})_{i\in[m]}$ and $\mathsf{dig}$, compute a list $\mathcal{L}_1$ in the same manner as $\mathsf{Output}'$.
6. If there exists $(y, \sigma) \in \mathcal{L}_1$ such that $y \neq D_x$, then output $((y, x), \sigma)$. Otherwise, output $\perp$.

If the event $\overline{\mathsf{BAD}_1} \wedge \mathsf{BAD}_2$ happens at the end of the experiment $\mathrm{EXPT}^{\text{LD-}s}_{\Pi',\mathcal{A}'}(\lambda)$, it means that a list $\mathcal{L}_1$ computed by a client contains $(y, \sigma)$ such that $y \neq D_x$ and $\Sigma.\mathsf{Verify}(\mathsf{pk}, (y, x), \sigma) = 1$. Since $\mathcal{A}_\Sigma$ internally executes $\mathrm{EXPT}^{\text{LD-}s}_{\Pi',\mathcal{A}'}(\lambda)$, the list $\mathcal{L}_1$ computed at Step 5 of $\mathcal{A}_\Sigma$ is the same as the one computed by the client in $\mathrm{EXPT}^{\text{LD-}s}_{\Pi',\mathcal{A}'}(\lambda)$. Therefore, the probability that $\overline{\mathsf{BAD}_1} \wedge \mathsf{BAD}_2$ happens is at most the probability that $\mathcal{A}_\Sigma$ outputs a message-signature pair $((y, x), \sigma)$ such that $(y, x) \notin \{(D_i, i)\}_{i\in[N]}$ and $\Sigma.\mathsf{Verify}(\mathsf{pk}, (y, x), \sigma) = 1$, which is negligible because of the unforgeability of $\Sigma$. □

## G.3 Details on the Construction Based on Vector Commitment

First, we define *updatable* PIR with a trusted setup. Our formalization is inspired by [44] but is not exactly the same as theirs in that our definition aims at unique decoding while they only considered verifiability only.

**Definition G.6.** *An $\ell$-round $m$-server* updatable PIR protocol with a trusted setup *is a tuple of polynomial-time algorithms*

$$\Pi = (\mathsf{Setup}, \mathsf{Query}, \mathsf{Answer}, \mathsf{Output}, \mathsf{Update}, \mathsf{ServerUpdate}),$$

*where:*

- $\mathsf{Setup}(1^\lambda, D) \to (\mathsf{pp}, \widetilde{D}, \mathsf{dig}, \mathsf{aux})$*:* $\mathsf{Setup}$ *is a randomized algorithm that takes a security parameter $\lambda$ and a database $D \in \mathcal{X}^N$ as input, and outputs public parameters* $\mathsf{pp}$*, an augmented database $\widetilde{D}$, a digest* $\mathsf{dig}$*, and auxiliary information* $\mathsf{aux}$*. We assume that* $\mathsf{pp}$ *is a common input to all the other algorithms. For notational simplicity, we omit* $\mathsf{pp}$ *from the inputs of the other algorithms.*
- $\mathsf{Query}(x, \mathsf{st}^{(j-1)}, (\mathsf{ans}_i^{(j-1)})_{i\in[m]}) \to ((\mathsf{que}_i^{(j)})_{i\in[m]}, \mathsf{st}^{(j)})$*:* $\mathsf{Query}$ *is a randomized algorithm that takes an index $x \in [N]$, a state $\mathsf{st}^{(j-1)}$ and answers $(\mathsf{ans}_i^{(j-1)})_{i\in[m]}$ in round $j-1$ as input, and outputs queries $(\mathsf{que}_i^{(j)})_{i\in[m]}$ and a state $\mathsf{st}^{(j)}$ in round $j$, where we define $\mathsf{st}^{(0)}$, $\mathsf{ans}_i^{(0)}$ as the empty string;*
- $\mathsf{Answer}(\widetilde{D}, \mathsf{que}_i^{(j)}) \to \mathsf{ans}_i^{(j)}$*:* $\mathsf{Answer}$ *is a deterministic algorithm that takes an augmented database $\widetilde{D}$ and a query $\mathsf{que}_i^{(j)}$ in round $j$ as input, and outputs an answer $\mathsf{ans}_i^{(j)}$ in round $j$;*
- $\mathsf{Output}(\mathsf{st}^{(\ell)}, (\mathsf{ans}_i^{(\ell)})_{i\in[m]}, \mathsf{dig}) \to y$*:* $\mathsf{Output}$ *is a deterministic algorithm that takes a state $\mathsf{st}^{(\ell)}$, answers $(\mathsf{ans}_i^{(\ell)})_{i\in[m]}$ in round $\ell$ and a digest* $\mathsf{dig}$ *as input, and outputs $y \in \mathcal{X}$;*

- $\mathsf{Update}((x, y'), \mathsf{aux}) \to (\mathsf{upd}, \mathsf{dig}', \mathsf{aux}')$*:* $\mathsf{Update}$ *is an algorithm that takes an index* $x \in [N]$ *and a new data item* $y' \in \mathcal{X}$ *as input, and outputs update information* $\mathsf{upd}$*, a new digest* $\mathsf{dig}'$ *and new auxiliary information* $\mathsf{aux}'$*;*
- $\mathsf{ServerUpdate}(\widetilde{D}, \mathsf{upd}) \to \widetilde{D}'$*:* $\mathsf{ServerUpdate}$ *is an algorithm that takes an augmented database* $\widetilde{D}$ *and update information* $\mathsf{upd}$ *as input, and outputs a new augmented database* $\widetilde{D}'$*;*

*such that* $\Pi$ *satisfies the following property:*

**Correctness.** *For any* $\lambda \in \mathbb{N}$*, any* $D = (D_1, \ldots, D_N) \in \mathcal{X}^N$*, any* $x \in [N]$*, any* $U \in \mathbb{N}$ *and any sequence of updates* $(x^{\langle u \rangle}, y^{\langle u \rangle})_{u \in [U]}$*, it holds that*

$$\Pr\left[\mathsf{Output}(\mathsf{st}^{(\ell)}, (\mathsf{ans}_i^{(\ell)})_{i \in [m]}), \mathsf{dig}^{\langle U \rangle}) = D_x^{\langle U \rangle}\right] = 1,$$

*where* $D_x^{\langle u \rangle}$ *is the* $x$*-th element of the database obtained by the updates* $(x^{\langle u \rangle}, y^{\langle u \rangle})_{u \in [U]}$ *and*

1. $(\mathsf{pp}, \widetilde{D}^{\langle 0 \rangle}, \mathsf{dig}^{\langle 0 \rangle}, \mathsf{aux}^{\langle 0 \rangle}) \leftarrow \mathsf{Setup}(1^\lambda, D^{\langle 0 \rangle})$.
2. *For each* $u \in [U]$*,*

$$(\mathsf{upd}^{\langle u \rangle}, \mathsf{dig}^{\langle u \rangle}, \mathsf{aux}^{\langle u \rangle}) \leftarrow \mathsf{Update}((x^{\langle u \rangle}, y^{\langle u \rangle}), \mathsf{aux}^{\langle u-1 \rangle}),$$
$$\widetilde{D}^{\langle u \rangle} \leftarrow \mathsf{ServerUpdate}(\widetilde{D}^{\langle u-1 \rangle}, \mathsf{upd}^{\langle u \rangle}).$$

3. *For each* $j \in [\ell]$ *and* $i \in [m]$*,*

$$((\mathsf{que}_i^{(j)})_{i \in [m]}, \mathsf{st}^{(j)}) \leftarrow \mathsf{Query}(x, \mathsf{st}^{(j-1)}, (\mathsf{ans}_i^{(j-1)})_{i \in [m]}),$$
$$\mathsf{ans}_i^{(j)} \leftarrow \mathsf{Answer}(\widetilde{D}^{\langle U \rangle}, \mathsf{que}_i^{(j)}).$$

*Define* the update cost *of* $\Pi$ *as* $|\mathsf{dig}'| + |\mathsf{upd}|$*, where* $|\mathsf{dig}'|$ *and* $\mathsf{upd}$ *are the bit-lengths of a updated digest* $\mathsf{dig}'$ *and update information* $\mathsf{upd}$ *output by* $\mathsf{Update}$*.*

We define the privacy of updatable PIR protocol with a trusted setup.

**Definition G.7.** *Let* $\Pi = (\mathsf{Setup}, \mathsf{Query}, \mathsf{Answer}, \mathsf{Output}, \mathsf{Update}, \mathsf{ServerUpdate})$ *be an* $\ell$*-round* $m$*-server updatable PIR protocol with a trusted setup. We say that* $\Pi$ *is* $t$-private *if it satisfies the following requirements:*

**Privacy.** *For any stateful PPT algorithm* $\mathcal{A}$*, the distributions of* $\mathsf{Priv}^0_{\Pi,\mathcal{A}}(\lambda)$ *and* $\mathsf{Priv}^1_{\Pi,\mathcal{A}}(\lambda)$ *are computatioanlly indistinguishable, where for* $\alpha \in \{0, 1\}$*,* $\mathsf{Priv}^\alpha_{\Pi,\mathcal{A}}(\lambda)$ *is the output* $\alpha'$ *of* $\mathcal{A}$ *in the following experiment* $\mathrm{EXPT}^{\text{Priv-s-u},\alpha}_{\Pi,\mathcal{A}}(\lambda)$*:*

1. $\mathcal{A}(1^\lambda)$ *outputs* $(D^{\langle 0 \rangle}, T)$ *such that* $D^{\langle 0 \rangle} \in \mathcal{X}^N$ *and* $T \subseteq [m]$ *is of size at most* $t$*.*
2. *Let* $(\mathsf{pp}, \widetilde{D}^{\langle 0 \rangle}, \mathsf{dig}^{\langle 0 \rangle}, \mathsf{aux}^{\langle 0 \rangle}) \leftarrow \mathsf{Setup}(1^\lambda, D^{\langle 0 \rangle})$ *and give* $\mathsf{pp}, \widetilde{D}^{\langle 0 \rangle}$ *to* $\mathcal{A}$*.*
3. *For a polynomial function* $U = U(\lambda)$ *and each* $u \in [U]$*, iterate the following:*
   (a) $\mathcal{A}$ *outputs* $(x^{\langle u \rangle}, y^{\langle u \rangle})$ *where* $x^{\langle u \rangle} \in [N]$ *and* $y^{\langle u \rangle} \in \mathcal{X}$*.*
   (b) *Let* $(\mathsf{upd}^{\langle u \rangle}, \mathsf{dig}^{\langle u \rangle}, \mathsf{aux}^{\langle u \rangle}) \leftarrow \mathsf{Update}((x^{\langle u \rangle}, y^{\langle u \rangle}), \mathsf{aux}^{\langle u-1 \rangle})$ *and* $\widetilde{D}^{\langle u \rangle} \leftarrow \mathsf{ServerUpdate}(\widetilde{D}^{\langle u-1 \rangle}, \mathsf{upd}^{\langle u \rangle})$*. Give* $\widetilde{D}^{\langle u \rangle}$ *to* $\mathcal{A}$*.*
4. $\mathcal{A}$ *outputs* $(x_0, x_1, u)$*, where* $x_0, x_1 \in [N]$ *and* $u \in \{0, 1, \ldots, U\}$*.*
5. *For each* $j = 1, 2, \ldots, \ell$*,*
   (a) *Let* $((\mathsf{que}_i^{(j)})_{i \in [m]}, \mathsf{st}^{(j)}) \leftarrow \mathsf{Query}(x_\alpha, \mathsf{st}^{(j-1)}, (\mathsf{ans}_i^{(j-1)})_{i \in [m]})$ *and give* $(\mathsf{que}_i^{(j)})_{i \in T}$ *to* $\mathcal{A}$*.*
   (b) *If* $j < \ell$*,* $\mathcal{A}$ *outputs* $(\mathsf{ans}_i^{(j)})_{i \in T}$*. Let* $\mathsf{ans}_i^{(j)} \leftarrow \mathsf{Answer}(\widetilde{D}^{\langle u \rangle}, \mathsf{que}_i^{(j)})$ *for* $i \notin T$*.*
   (c) *If* $j = \ell$*,* $\mathcal{A}$ *outputs a bit* $\alpha' \in \{0, 1\}$*.*

We extend the notion of LD-PIR to the setting of updatable PIR with a trusted setup.

**Definition G.8.** *Let* $\Pi = (\mathsf{Setup}, \mathsf{Query}, \mathsf{Answer}, \mathsf{Output}, \mathsf{Update}, \mathsf{ServerUpdate})$ *be an* $\ell$*-round* $m$*-server updatable PIR protocol with a trusted setup. We say that* $\Pi$ *is* $(b, L)$-list-decodable *if* $\mathsf{Output}$ *is allowed to output a subset* $\mathcal{L} \subseteq \mathcal{X}$ *and* $\Pi$ *satisfies the following requirements:*

**List decoding.** *There exists a negligible function* $\epsilon = \epsilon(\lambda)$ *such that for any* $\lambda \in \mathbb{N}$ *and any stateful PPT algorithm* $\mathcal{A}$*, it holds that* $\Pr[\mathsf{LD}_{\Pi,\mathcal{A}}(\lambda) = 1] \leq \epsilon$*, where* $\mathsf{LD}_{\Pi,\mathcal{A}}(\lambda)$ *is the output of the following experiment* $\mathrm{EXPT}^{\text{LD-s-u}}_{\Pi,\mathcal{A}}(\lambda)$*:*

1. $\mathcal{A}(1^\lambda)$ *outputs* $(D^{\langle 0\rangle}, T)$ *such that* $D^{\langle 0\rangle} \in \mathcal{X}^N$ *and* $T \subseteq [m]$ *is of size at most* $t$*.*
2. *Let* $(\mathsf{pp}, \widetilde{D}^{\langle 0\rangle}, \mathsf{dig}^{\langle 0\rangle}, \mathsf{aux}^{\langle 0\rangle}) \leftarrow \mathsf{Setup}(1^\lambda, D^{\langle 0\rangle})$ *and give* $\mathsf{pp}, \widetilde{D}^{\langle 0\rangle}$ *to* $\mathcal{A}$*.*
3. *For a polynomial function* $U = U(\lambda)$ *and each* $u \in [U]$*, iterate the following:*
   (a) $\mathcal{A}$ *outputs* $(x^{\langle u\rangle}, y^{\langle u\rangle})$ *where* $x^{\langle u\rangle} \in [N]$ *and* $y^{\langle u\rangle} \in \mathcal{X}$*.*
   (b) *Let* $(\mathsf{upd}^{\langle u\rangle}, \mathsf{dig}^{\langle u\rangle}, \mathsf{aux}^{\langle u\rangle}) \leftarrow \mathsf{Update}((x^{\langle u\rangle}, y^{\langle u\rangle}), \mathsf{aux}^{\langle u-1\rangle})$ *and* $\widetilde{D}^{\langle u\rangle} \leftarrow \mathsf{ServerUpdate}(\widetilde{D}^{\langle u-1\rangle}, \mathsf{upd}^{\langle u\rangle})$*. Give* $\widetilde{D}^{\langle u\rangle}$ *to* $\mathcal{A}$*.*
4. $\mathcal{A}$ *outputs* $(x, u)$*, where* $x \in [N]$ *and* $u \in \{0, 1, \ldots, U\}$*.*
5. *For each* $j = 1, 2, \ldots, \ell$*,*
   (a) *Let* $((\mathsf{que}_i^{(j)})_{i\in[m]}, \mathsf{st}^{(j)}) \leftarrow \mathsf{Query}(x, \mathsf{st}^{(j-1)}, (\mathsf{ans}_i^{(j-1)})_{i\in[m]})$ *and give* $(\mathsf{que}_i^{(j)})_{i\in T}$ *to* $\mathcal{A}$*.*
   (b) $\mathcal{A}$ *outputs* $(\mathsf{ans}_i^{(j)})_{i\in T}$*. Let* $\mathsf{ans}_i^{(j)} \leftarrow \mathsf{Answer}(\widetilde{D}^{\langle u\rangle}, \mathsf{que}_i^{(j)})$ *for* $i \notin T$*.*
6. *Let* $\mathcal{L} \leftarrow \mathsf{Output}(\mathsf{st}^{(\ell)}, (\mathsf{ans}_i^{(\ell)})_{i\in[m]}, \mathsf{dig}^{\langle u\rangle})$*. Return* $0$ *if* $D_x^{\langle u\rangle} \in \mathcal{L}$ *and* $|\mathcal{L}| \leq L$*, and otherwise return* $1$*, where* $D_x^{\langle u\rangle}$ *is the* $x$*-th element of the database* $D^{\langle u\rangle}$ *after* $u$ *updates.*

We recall the notion of vector commitment [9, 41].

**Definition G.9** (Vector commitment)**.** *A vector commitment scheme is a tuple of polynomial-time algorithms*

$$\Sigma = (\mathsf{Gen}, \mathsf{Commit}, \mathsf{Open}, \mathsf{Verify}, \mathsf{Update}, \mathsf{ProofUpdate}),$$

*where:*

- $\mathsf{Gen}(1^\lambda, N) \to \mathsf{pp}$*:* $\mathsf{Gen}$ *is an algorithm that takes a security parameter* $\lambda$ *and the size* $N$ *of a committed vector as input and outputs a public parameter*$\mathsf{pp}$*. We assume that* $\mathsf{pp}$ *is a common input to all the other algorithms. For notational simplicity, we omit* $\mathsf{pp}$ *from the inputs of the other algorithms.;*
- $\mathsf{Commit}(D) \to (C, \mathsf{data})$*:* $\mathsf{Commit}$ *is an algorithm that takes a message vector* $D$ *as input and outputs a commitment* $C$ *and auxiliary information* $\mathsf{data}$ *used to produce the opening proofs of the messages;*
- $\mathsf{Open}((y, x), \mathsf{data}) \to \pi$*:* $\mathsf{Open}$ *is an algorithm that takes a message* $y$*, an index* $x$ *and auxiliary information* $\mathsf{data}$*, and outputs a proof* $\pi$ *that* $y$ *is the committed message at position* $x$*;*
- $\mathsf{Verify}(C, (y, x), \pi) \to b$*:* $\mathsf{Verify}$ *is an algorithm that takes a commitment* $C$*, a message* $y$*, an index* $x$ *and a proof* $\pi$*, and outputs a bit* $b$*;*
- $\mathsf{Update}(C, x, y, y', \mathsf{data}) \to (C', U, \mathsf{data})$*:* $\mathsf{Update}$ *is an algorithm that takes an old commitment* $C$*, an index* $x$*, an old message* $y$*, a new message* $y'$ *and auxiliary information* $\mathsf{data}$*, and outputs a new commitment* $C'$*, update information* $U$ *and updated auxiliary information* $\mathsf{data}$*;*
- $\mathsf{ProofUpdate}(C', \pi, (y', x), U) \to \pi'$*:* $\mathsf{ProofUpdate}$ *is an algorithm that takes an updated commitment* $C'$*, an old proof* $\pi$*, a new message* $y'$*, an index* $x$*, and update information* $U$*, and outputs a new proof* $\pi'$ *with respect to* $C'$ *that* $y'$ *is the committed message at position* $x$*;*

*satisfying the following properties:*

**Correctness.** *For any* $\lambda \in \mathbb{N}$ *and any polynomial function* $N = N(\lambda)$*, it holds with probability* $1$ *that* $\mathsf{Verify}(C, (D_x, x), \pi) = 1$ *for any* $x \in [N]$*, where* $\mathsf{pp} \leftarrow \mathsf{Gen}(1^\lambda, N)$*,* $C$ *is a commitment on a message vector* $D = (D_1, \ldots, D_N)$ *of size* $N$ *obtained by running* $\mathsf{Commit}$ *possibly followed by a sequence of updates, and* $\pi$ *is a proof for position* $x$ *generated by* $\mathsf{Open}$ *or* $\mathsf{ProofUpdate}$*;*

**Position binding.** *There exists a negligible function* $\mathsf{negl}(\lambda)$ *such that for any stateful PPT algorithm* $\mathcal{A}$*, any* $\lambda \in \mathbb{N}$ *and any polynomial function* $N = N(\lambda)$*, it holds that* $\Pr[\mathsf{PB}_{\Sigma,\mathcal{A}}(\lambda) = 1] \leq \mathsf{negl}(\lambda)$*, where* $\mathsf{PB}_{\Sigma,\mathcal{A}}(\lambda)$ *is the output of the following experiment:*

1. *Let* $\mathsf{pp} \leftarrow \mathsf{Gen}(1^\lambda, N)$ *and give* $\mathsf{pp}$ *to* $\mathcal{A}$*.*
2. $\mathcal{A}$ *outputs* $(C, y, y', x, \pi, \pi')$*. Return* 1 *if* $y \neq y'$ *and* $\mathsf{Verify}(C, (y, x), \pi) = \mathsf{Verify}(C, (y', x), \pi') = 1$ *and otherwise return* 0*.*

We show our construction of $(b, 1)$-LD-PIR with a trusted setup based on vector commitment.

**Theorem G.10.** *Let* $\Pi$ *be a* $t$*-private* $\ell$*-round* $m$*-server* $(b, L)$*-LD-PIR protocol. Let* $\Sigma$ *be a vector commitment scheme. Then, there exists a* $t$*-private* $\ell$*-round* $m$*-server updatable* $(b, 1)$*-LD-PIR protocol* $\Pi'$ *with a trusted setup such that* $\mathrm{Comm}(\Pi') = |\pi| \cdot \mathrm{Comm}(\Pi)$*, the setup cost is* $|C|$*, and the update cost is* $|C| + |U| + \log|\mathcal{X}| + \log N$*, where* $|\pi|$*,* $|C|$ *and* $|U|$ *are the bit-lengths of a proof, a commitment and update information of* $\Sigma$*, respectively.*

*Proof.* We describe the construction of $\Pi'$ in Figure 11.

**Complexity.** By construction, the communication complexity of $\Pi'$ is $\pi$ times larger than $\Pi$. The setup cost is the bit-length of a commitment $C$ of $\Sigma$. Furthermore, it is clear that the update cost is the sum of the bit-lengths of a new commitment, update information of $\Sigma$, a new data item $y \in \mathcal{X}$ and a corresponding position $x \in [N]$.

**Privacy.** We reduce the privacy of $\Pi'$ to that of $\Pi$. Let $\mathcal{A}'$ be an adversary against the privacy of $\Pi'$. We construct an adversary $\mathcal{A}$ against the privacy of $\Pi$ as follows.

$\mathcal{A}$

**Setup.** On input $1^\lambda$,

1. Run $(D^{\langle 0\rangle}, T) \leftarrow \mathcal{A}'(1^\lambda)$, where $D^{\langle 0\rangle} \in \mathcal{X}^N$ and $T$ is a subset of $[m]$ of size $t$.
2. Run $(\mathsf{pp}, \widetilde{D}^{\langle 0\rangle}, \mathsf{dig}^{\langle 0\rangle}, \mathsf{aux}^{\langle 0\rangle}) \leftarrow \mathsf{Setup}'(1^\lambda, D^{\langle 0\rangle})$.
3. For a polynomial function $U = U(\lambda)$ and each $u \in [U]$, iterate the following:
   (a) Run $(x^{\langle u\rangle}, y^{\langle u\rangle}) \leftarrow \mathcal{A}'$, where $x^{\langle u\rangle} \in [N]$ and $y^{\langle u\rangle} \in \mathcal{X}$.
   (b) Let $(\mathsf{upd}^{\langle u\rangle}, \mathsf{dig}^{\langle u\rangle}, \mathsf{aux}^{\langle u\rangle}) \leftarrow \mathsf{Update}((x^{\langle u\rangle}, y^{\langle u\rangle}), \mathsf{aux}^{\langle u-1\rangle})$. Give $\mathsf{upd}^{\langle u\rangle}$ to $\mathcal{A}'$.
   (c) Let $\widetilde{D}^{\langle u\rangle} \leftarrow \mathsf{ServerUpdate}(\widetilde{D}^{\langle u-1\rangle}, \mathsf{upd}^{\langle u\rangle})$.
4. Run $(x_0, x_1, u) \leftarrow \mathcal{A}'$, where $x_0, x_1 \in [N]$ and $u \in \{0, 1, \ldots, U\}$.
5. Output $(x_0, x_1, \widetilde{D}^{\langle u\rangle}, T)$.

**Protocol.** During the execution of $\Pi$, $\mathcal{A}$ emulates the behavior of $\mathcal{A}'$. More precisely, to compute an answer of $\mathsf{S}_i$ such that $i \in T$, $\mathcal{A}$ relays the answer of $\mathsf{S}_i$ computed by $\mathcal{A}'$.

**Output.** To output a guess bit $\alpha'$, $\mathcal{A}$ relays the bit output by $\mathcal{A}'$.

It can be seen that $\mathsf{Priv}^{\alpha}_{\Pi,\mathcal{A}}(\lambda) = \mathsf{Priv}^{\alpha}_{\Pi',\mathcal{A}'}(\lambda)$ for $\alpha \in \{0, 1\}$, since the information that $\mathcal{A}$ gives to $\mathcal{A}'$ is exactly the same as the one that $\mathcal{A}'$ sees during the experiment $\mathrm{EXPT}^{\text{Priv-s-u},\alpha}_{\Pi',\mathcal{A}'}(\lambda)$. Therefore, the distributions of $\mathsf{Priv}^{0}_{\Pi',\mathcal{A}'}(\lambda)$ and $\mathsf{Priv}^{1}_{\Pi',\mathcal{A}'}(\lambda)$ are statistically (resp. computationally) indistinguishable if $\mathsf{Priv}^{0}_{\Pi,\mathcal{A}}(\lambda)$ and $\mathsf{Priv}^{1}_{\Pi,\mathcal{A}}(\lambda)$ are statistically (resp. computationally) indistinguishable.

**Unique decoding.** We reduce the $(b, 1)$-list decoding property of $\Pi'$ to the $(b, L)$-list decoding property of $\Pi$ and the position binding property of $\Sigma$. Let $\mathcal{A}'$ be an adversary against the $(b, 1)$-list decoding property of $\Pi'$.

Define $\mathsf{BAD}_1$ as an event that $D_x \notin \mathcal{L}$ during the experiment $\mathrm{EXPT}^{\text{LD-s-u}}_{\Pi',\mathcal{A}'}(\lambda)$. Also, define $\mathsf{BAD}_2$ as an event that $|\mathcal{L}| > 1$ during the experiment $\mathrm{EXPT}^{\text{LD-s-u}}_{\Pi',\mathcal{A}'}(\lambda)$. Since $\Pr[\mathsf{LD}_{\Pi',\mathcal{A}'}(\lambda) = 1] = \Pr[\mathsf{BAD}_1 \vee \mathsf{BAD}_2] \leq$

**Notations.**

- Let $\Sigma = (\mathsf{Gen}, \mathsf{Commit}, \mathsf{Open}, \mathsf{Verify}, \mathsf{Update}, \mathsf{ProofUpdate})$ be a vector commitment scheme.
- Let $\Pi = (\mathsf{Query}, \mathsf{Answer}, \mathsf{Output})$ be a $t$-private $\ell$-round $m$-server $(b, L)$-LD-PIR protocol.

$\mathsf{Setup}'(1^\lambda, D)$**.** Given a security parameter $\lambda$ and a database $D \in \mathcal{X}^N$:

1. $\mathsf{pp} \leftarrow \Sigma.\mathsf{Gen}(1^\lambda, N)$.
2. $(C, \mathsf{data}) \leftarrow \Sigma.\mathsf{Commit}(D)$.
3. For each $x \in [N]$, let $\pi_x \leftarrow \Sigma.\mathsf{Open}((D_x, x), \mathsf{data})$.
4. Output a public parameter $\mathsf{pp}$, an augmented database $\widetilde{D} = \{(\widetilde{D}_1, \ldots, \widetilde{D}_N), C\}$ where $\widetilde{D}_x = (D_x, \pi_x)$, a digest $\mathsf{dig} = C$, $\mathsf{aux} = \{D, \mathsf{data}, C\}$.

$\mathsf{Query}'(x, \mathsf{st}^{(j-1)}, (\mathsf{ans}_i^{(j-1)})_{i\in[m]})$**.** The same as $\mathsf{Query}(x, \mathsf{st}^{(j-1)}, (\mathsf{ans}_i^{(j-1)})_{i\in[m]})$.

$\mathsf{Answer}'(\widetilde{D}, \mathsf{que}_i^{(j)})$**.** The same as $\mathsf{Answer}'(D, \mathsf{que}_i^{(j)})$ except that an answer is computed on the augmented one $\widetilde{D}$ instead of $D$.

$\mathsf{Output}'(\mathsf{st}^{(\ell)}, (\mathsf{ans}_i^{(\ell)})_{i\in[m]}, \mathsf{dig})$**.** Given a state $\mathsf{st}^{(\ell)}$, answers $\mathsf{ans}_i^{(\ell)}$ and a digest $\mathsf{dig} = C$:

1. Compute a list
$$\mathcal{L}_0 \leftarrow \mathsf{Output}(\mathsf{st}^{(\ell)}, (\mathsf{ans}_i^{(\ell)})_{i\in[m]}).$$
2. Let
$$\mathcal{L}_1 = \{(y, \pi) \in \mathcal{L}_0 : \Sigma.\mathsf{Verify}(C, (y, x), \pi) = 1\}.$$
3. Output $\mathcal{L} = \{y : (y, \pi) \in \mathcal{L}_1\}$.

$\mathsf{Update}'((y', x), \mathsf{aux})$**.** Given a new data item $y'$, an index $x$ and auxiliary information $\mathsf{aux}$:

1. Let $y$ be the $x$-th element of the current database $D$ and $C$ be a commitment of $D$, which are included in $\mathsf{aux}$.
2. Let $(C', U, \mathsf{data}') \leftarrow \Sigma.\mathsf{Update}(C, x, y, y', \mathsf{data})$.
3. Output update information $\mathsf{upd} = \{C', U, (y', x)\}$, a new digest $\mathsf{dig}' = C'$, and auxiliary information $\mathsf{aux}' = \{C', D', \mathsf{data}'\}$, where $D'$ is an updated database.

$\mathsf{ServerUpdate}'(\widetilde{D}, \mathsf{upd})$**.** Given an augmented database $\widetilde{D}$ and update information $\mathsf{upd} = \{C', U, (y', x)\}$:

1. For each $i \in [N]$, let $\pi_i' \leftarrow \Sigma.\mathsf{ProofUpdate}(C', \pi_i, (y', x), U)$, where $C$ and $\pi_i$ are an old commitment and an old proof included in $\widetilde{D}$, respectively.
2. Output an augmented database $(D_i', \pi_i')_{i\in[N]}$ along with a new commitment $C'$ as $\widetilde{D}'$, where $D_i' = D_i$ for $i \neq x$ and $D_i' = y'$ for $i = x$.

Figure 12: A construction of $\Pi'$ based on a vector commitment scheme

$\Pr[\mathsf{BAD}_1] + \Pr\left[\overline{\mathsf{BAD}_1} \wedge \mathsf{BAD}_2\right]$, it is sufficient to show that

$$\Pr[\mathsf{BAD}_1] \leq \mathsf{negl}(\lambda)\,, \tag{18}$$

$$\Pr\left[\overline{\mathsf{BAD}_1} \wedge \mathsf{BAD}_2\right] \leq \mathsf{negl}(\lambda) \tag{19}$$

for a negligible function $\mathsf{negl}(\lambda)$.

To prove (18), we construct an adversary $\mathcal{A}$ against the $(b, L)$-list decoding property of $\Pi$ as follows.

$\mathcal{A}$

**Setup.** On input $1^\lambda$,

1. Run $(D^{\langle 0\rangle}, T) \leftarrow \mathcal{A}'(1^\lambda)$, where $D^{\langle 0\rangle} \in \mathcal{X}^N$ and $T$ is a subset of $[m]$ of size $b$.
2. Run $(\mathsf{pp}, \widetilde{D}^{\langle 0\rangle}, \mathsf{dig}^{\langle 0\rangle}, \mathsf{aux}^{\langle 0\rangle}) \leftarrow \mathsf{Setup}'(1^\lambda, D^{\langle 0\rangle})$.
3. For a polynomial function $U = U(\lambda)$ and each $u \in [U]$, iterate the following:
   (a) Run $(x^{\langle u\rangle}, y^{\langle u\rangle}) \leftarrow \mathcal{A}'$, where $x^{\langle u\rangle} \in [N]$ and $y^{\langle u\rangle} \in \mathcal{X}$.
   (b) Let $(\mathsf{upd}^{\langle u\rangle}, \mathsf{dig}^{\langle u\rangle}, \mathsf{aux}^{\langle u\rangle}) \leftarrow \mathsf{Update}((x^{\langle u\rangle}, y^{\langle u\rangle}), \mathsf{aux}^{\langle u-1\rangle})$. Give $\mathsf{upd}^{\langle u\rangle}$ to $\mathcal{A}'$.
   (c) Let $\widetilde{D}^{\langle u\rangle} \leftarrow \mathsf{ServerUpdate}(\widetilde{D}^{\langle u-1\rangle}, \mathsf{upd}^{\langle u\rangle})$.
4. Run $(x, u) \leftarrow \mathcal{A}'$, where $x_0, x_1 \in [N]$ and $u \in \{0, 1, \ldots, U\}$.
5. Output $(x, \widetilde{D}^{\langle u\rangle}, T)$.

**Protocol.** During the execution of $\Pi$, $\mathcal{A}$ emulates the behavior of $\mathcal{A}'$. More precisely, to compute an answer of $\mathsf{S}_i$ such that $i \in T$, $\mathcal{A}$ relays the answer of $\mathsf{S}_i$ computed by $\mathcal{A}'$.

Consider the experiment $\mathrm{EXPT}^{\mathrm{LD}}_{\Pi,\mathcal{A}}(\lambda)$ defined in Definition A.2. Let $\mathcal{L}_0$ be a list output by a client in $\mathrm{EXPT}^{\mathrm{LD}}_{\Pi,\mathcal{A}}(\lambda)$. It follows from the constructions of $\Pi$ and $\mathcal{A}$ that the list $\mathcal{L}_0$ computed at the end of $\mathrm{EXPT}^{\mathrm{LD}}_{\Pi,\mathcal{A}}(\lambda)$ is the same as $\mathcal{L}_0$ obtained at Step 1 of $\mathsf{Output}'$ in $\mathrm{EXPT}^{\text{LD-s-u}}_{\Pi',\mathcal{A}'}(\lambda)$. If $\mathsf{BAD}_1$ happens, then $(D^{\langle u\rangle}_x, \pi_x)$ is not included in $\mathcal{L}_0$ in $\mathrm{EXPT}^{\text{LD-s-u}}_{\Pi',\mathcal{A}'}(\lambda)$, which implies that $\mathsf{LD}_{\Pi,\mathcal{A}}(\lambda) = 1$. Since $\Pi$ is $(b, L)$-list-decodable, we have that $\Pr[\mathsf{BAD}_1]$ is negligible.

To prove (19), we construct an adversary $\mathcal{A}_\Sigma$ against the position binding property of $\Sigma$.

$\mathcal{A}_\Sigma$

1. On input $1^\lambda$, run $(D^{\langle 0\rangle}, T) \leftarrow \mathcal{A}'(1^\lambda)$, where $D^{\langle 0\rangle} \in \mathcal{X}^N$ and $T$ is a subset of $[m]$ of size $b$.
2. Simulate the output of $\mathsf{Setup}'(1^\lambda, D^{\langle 0\rangle})$. That is, upon receiving a public parameter $\mathsf{pp}$ of $\Sigma$ in the security game, compute $(\widetilde{D}^{\langle 0\rangle}, \mathsf{dig}^{\langle 0\rangle}, \mathsf{aux}^{\langle 0\rangle})$ in the same manner as $\mathsf{Setup}'$. Then, give $\widetilde{D}^{\langle 0\rangle}$ to $\mathcal{A}'$.
3. For a polynomial function $U = U(\lambda)$ and each $u \in [U]$, iterate the following:
   (a) Run $(x^{\langle u\rangle}, y^{\langle u\rangle}) \leftarrow \mathcal{A}'$, where $x^{\langle u\rangle} \in [N]$ and $y^{\langle u\rangle} \in \mathcal{X}$.
   (b) Let $(\mathsf{upd}^{\langle u\rangle}, \mathsf{dig}^{\langle u\rangle}, \mathsf{aux}^{\langle u\rangle}) \leftarrow \mathsf{Update}((x^{\langle u\rangle}, y^{\langle u\rangle}), \mathsf{aux}^{\langle u-1\rangle})$. Give $\mathsf{upd}^{\langle u\rangle}$ to $\mathcal{A}'$.
   (c) Let $\widetilde{D}^{\langle u\rangle} \leftarrow \mathsf{ServerUpdate}(\widetilde{D}^{\langle u-1\rangle}, \mathsf{upd}^{\langle u\rangle})$.
4. Run $(x, u) \leftarrow \mathcal{A}'$, where $x \in [N]$ and $u \in \{0, 1, \ldots, U\}$.
5. Locally simulate the execution of $\Pi'$ on a client input $x$. That is, for each $j = 1, 2, \ldots, \ell$,
   (a) Let $((\mathsf{que}_i^{(j)})_{i\in[m]}, \mathsf{st}^{(j)}) \leftarrow \mathsf{Query}'(x, \mathsf{st}^{(j-1)}, (\mathsf{ans}_i^{(j-1)})_{i\in[m]})$ and give $(\mathsf{que}_i^{(j)})_{i\in T}$ to $\mathcal{A}'$.
   (b) $\mathcal{A}'$ outputs $(\mathsf{ans}_i^{(j)})_{i\in T}$. Let $\mathsf{ans}_i^{(j)} \leftarrow \mathsf{Answer}'(\widetilde{D}^{\langle u\rangle}, \mathsf{que}_i^{(j)})$ for $i \notin T$.
6. Based on $\mathsf{st}^{(\ell)}$, $(\mathsf{ans}_i^{(\ell)})_{i\in[m]}$ and $\mathsf{dig}^{\langle u\rangle}$, compute a list $\mathcal{L}_1$ in the same manner as $\mathsf{Output}'$.
7. If there exists $(y, \pi) \in \mathcal{L}_1$ such that $y \neq D_x^{\langle u\rangle}$, then output $(C^{\langle u\rangle}, D_x^{\langle u\rangle}, y, x, \pi_x^{\langle u\rangle}, \pi)$. Otherwise, output $\perp$.

If the event $\overline{\mathsf{BAD}_1} \wedge \mathsf{BAD}_2$ happens at the end of the experiment $\mathrm{EXPT}^{\text{LD-s-u}}_{\Pi',\mathcal{A}'}(\lambda)$, it means that a list $\mathcal{L}_1$ computed by a client contains $(y, \pi)$ such that $y \neq D_x$ and $\Sigma.\mathsf{Verify}(C, (y, x), \pi) = 1$. Since $\mathcal{A}_\Sigma$ internally executes $\mathrm{EXPT}^{\text{LD-s-u}}_{\Pi',\mathcal{A}'}(\lambda)$, the list $\mathcal{L}_1$ computed at Step 6 of $\mathcal{A}_\Sigma$ is the same as the one computed by the client in $\mathrm{EXPT}^{\text{LD-s-u}}_{\Pi',\mathcal{A}'}(\lambda)$. Therefore, the probability that $\overline{\mathsf{BAD}_1} \wedge \mathsf{BAD}_2$ happens is at most the probability that $\mathcal{A}_\Sigma$ outputs $(C^{\langle u\rangle}; D_x^{\langle u\rangle}, y; x; \pi_x^{\langle u\rangle}, \pi)$ such that $D_x^{\langle u\rangle} \neq y$ and $\pi_x^{\langle u\rangle}$ (resp. $\pi$) is a valid proof that $D_x^{\langle u\rangle}$ (resp. $y$) is a committed message at position $x$ with respect to $C^{\langle u\rangle}$. The position-binding property of $\Sigma$ ensures that the probability is negligible. □

# H Details on LD-PIR Based on Polynomial Interpolation

## H.1 Preparation

We denote $[0..n] = \{0, 1, 2, \ldots, n\}$. For two vectors $\boldsymbol{x} = (x_i)_{i\in[\ell]}$ and $\boldsymbol{y} = (y_i)_{i\in[\ell]}$, we denote $\langle \boldsymbol{x}, \boldsymbol{y}\rangle = \sum_{i\in[\ell]} x_i y_i$. For a prime $p$, we denote the field of size $p$ by $\mathbb{F}_p$. Let $f \in \mathbb{F}_p[z_1, \ldots, z_\ell]$ be an $\ell$-variate polynomial over $\mathbb{F}_p$. We say that $f$ has degree $k$ if its total degree is at most $k$. Denote the partial derivative of $f$ with respect by $z_i$ as $\partial f/\partial z_i$. If $f$ is a univariate polynomial, the derivative of $f$ with respect to its unique variable is simply denoted by $f'$.

Hermite interpolation [39] is a technique to recover a polynomial $f(X)$ from its values $f(\alpha_i)$ and derivatives $f'(\alpha_i)$ on given points $\alpha_i$. Let $\alpha_1, \ldots, \alpha_\ell$ be $\ell$ pairwise distinct elements of $\mathbb{F}_p$. For each $j \in [\ell]$ and each $w \in \{0, 1\}$, let $y_{j,w} \in \mathbb{F}_p$. In [39], an explicit formula was shown for finding a unique polynomial $f(X) \in \mathbb{F}_p[X]$ such that

- $\deg f \leq 2\ell - 1$;
- $f(\alpha_i) = y_{j,0}$ and $f'(\alpha_i) = y_{j,1}$ for all $j \in [\ell]$.

In particular, if a degree-$(2\ell - 1)$ polynomial $f(X)$ satisfies $f(\alpha_i) = f'(\alpha_i) = 0$ for at least $\ell$ points $\alpha_i$, then we can conclude that $f(X)$ is the zero polynomial.

## H.2 List Decoding for Woodruff-Yekhanin's PIR

In Figure 13, we show the formal description of Woodruff-Yekhanin's PIR protocol [42], which is the one shown in Proposition F.2. Let $M$ be a parameter such that $\binom{M}{d} \geq N$, which can be chosen as $M = O(dN^{1/d})$. We choose a prime $p$ such that $p \geq \max\{m+1, |\mathcal{X}|\}$, which can be chosen as $p = O(m + |\mathcal{X}|)$. In particular, all databases $D \in \mathcal{X}^N$ can be embedded into $\mathbb{F}_p^N$. It can be easily verified that the communication complexity of the protocol is $O(mM \log p) = \widetilde{O}(N^{1/d} md \log |\mathcal{X}|)$.

To upgrade Woodruff-Yekhanin's protocol into LD-PIR, we apply the existing list-decoding algorithms [28, 31, 32] for (univariate) multiplicity codes. Since [28, 31, 32] provided only asymptotic analyses allowing the list size $L$ to grow, we provide an analysis focusing on a concretely small list size $L$.

By abuse of notation, we denote the Hamming distance between block vectors $\boldsymbol{x} = (\boldsymbol{x}_i)_{i\in[m]}$ and $\boldsymbol{y} = (\boldsymbol{y}_i)_{i\in[m]} \in (\mathbb{F}_p^2)^m$ by $\mathsf{dist}(\boldsymbol{x}, \boldsymbol{y}) = |\{i : \boldsymbol{x}_i \neq \boldsymbol{y}_i\}|$. We have a list decoding algorithm that on input $\boldsymbol{c} \in (\mathbb{F}_p^2)^m$, computes a list of all degree-$k$ polynomials $f$ such that the Hamming distance between $\boldsymbol{c}$ and $(f(\alpha_i), f'(\alpha_i))_{i\in[m]}$ is at most $b$.

**Theorem H.1.** *Assume that*

$$2m < \sum_{\nu=0}^{L} \max\{2(m-b) - \nu k,\, 0\}. \tag{20}$$

*There exists an algorithm such that on input $\boldsymbol{c} \in (\mathbb{F}_p^2)^m$, it outputs the list of all polynomials $f \in \mathbb{F}_p[X]$ such that $\deg f \leq k$ and $\mathsf{dist}(\boldsymbol{c}, (f(\alpha_i), f'(\alpha_i))_{i\in[m]}) \leq b$. Furthermore, the size of the list is at most $L$. The computation complexity is polynomial in $m$, $p$ and $L$.*

*Proof.* Consider the algorithm shown in Figure 14.

First, we show that it is possible to efficiently find $R_\nu \in \mathbb{F}_p[X]$ for $\nu \in [0..L]$ at Step 1 of the algorithm shown in Figure 14. Observe that the condition (21) gives $A := 2m$ linear equations in the coefficients of the $R_\nu$'s and that the total number of the unknown coefficients is $B := \sum_{\nu\in[0..L]} \max\{2(m-b) - \nu k, 0\}$. We then have $A < B$ from the assumption. Hence we can find a non-trivial solution satisfying the condition (21) in polynomial time in $m$, $L$, $k$ and $\log p$.

Next, we can find the list $\mathcal{F}$ of all degree-$d$ polynomials $f \in \mathbb{F}_p[X]$ such that $Y - f(X)$ divides $R(X, Y)$ as follows. Fix any irreducible polynomial $q(X) \in \mathbb{F}_p[X]$ of degree at least $k+1$, which can be found in advance by a deterministic algorithm in polynomial time in $k$ and $p$ [37]. Note that $\mathcal{F}$ is included in the list $\mathcal{G}$ of all polynomials $g \in \mathbb{F}_p[X]$ such that $R(X, g(X)) = 0 \bmod q(x)$ and $\deg g < \deg q$. We can obtain $\mathcal{F}$ from $\mathcal{G}$ by checking if $R(X, g(X))$ is the zero polynomial and $\deg g \leq k$ for each $g \in \mathcal{G}$. To obtain $\mathcal{G}$, we can view $R(X, Y) \bmod q(X)$ as a polynomial $Q(Y) \in \mathbb{E}[Y]$ with coefficients in the extension field $\mathbb{E} = \mathbb{F}_p[X]/(q(X))$. Then, obtaining $\mathcal{G}$ is reduced to finding roots of $Q(Y)$ in $\mathbb{E}$, which can also be solved by a deterministic algorithm in polynomial time in $k$ and $p$ [38, Chapter 1].

Finally, the proof completes if we show that $Y - f(X)$ divides $R(X, Y)$ for any degree-$k$ polynomial $f \in \mathbb{F}_p[X]$ such that $\mathsf{dist}(\boldsymbol{c}, (f(\alpha_i), f'(\alpha_i))_{i\in[m]}) \leq b$. It is sufficient to show that $P(X) := R(X, f(X)) \in \mathbb{F}_p[X]$ is the zero polynomial. Observe that $\deg P \leq \max_{\nu\in[0..L]}\{\nu k + \deg R_\nu\} < 2(m-b)$ since $\deg R_\nu \leq 2(m-b) - \nu k - 1$ for all $\nu \in [0..L]$. Let $i \in [m]$ be an index such that $f(\alpha_i) = c_{i0}$ and $f'(\alpha_i) = c_{i1}$. The condition (21) then implies that $P(\alpha_i) = \sum_{\nu=0}^{L} f(\alpha_i)^\nu R_\nu(\alpha_i) = \sum_{\nu=0}^{L} c_{i0}^\nu R_\nu(\alpha_i) = 0$ and $P'(\alpha_i) = \sum_{\nu=0}^{L}(f(\alpha_i)^\nu R'_\nu(\alpha_i) + \nu f(\alpha_i)^{\nu-1} f'(\alpha_i) R_\nu(\alpha_i)) = \sum_{\nu=0}^{L}(c_{i0}^\nu R'_\nu(\alpha_i) + \nu c_{i0}^{\nu-1} c_{i1} R_\nu(\alpha_i)) = 0$. Since there are at least $m - b$ such indices, we conclude that $P$ is the zero polynomial from Hermite interpolation. □

Finally, by applying the list decoding algorithm to Woodruff-Yekhanin's protocol, we can show the following theorem, which is a restatement of Proposition 5.1.

**Proposition H.2.** *For a parameter $d \in \mathbb{N}$, suppose that*

$$m > \frac{L+1}{L} b + \frac{L+1}{4} dt \text{ and } m \geq b + \frac{L}{2} dt. \tag{22}$$

*Then, there exists a $t$-private 1-round $m$-server $(b, L)$-LD-PIR protocol $\Pi$ such that $\mathrm{Comm}(\Pi) = \widetilde{O}(N^{1/d} md \log |\mathcal{X}|)$. The computation complexity of $\Pi$ is polynomial in $m$, $|\mathcal{X}|$ and $L$.*

**Notations.**

- Let $M \in \mathbb{N}$ be such that $\binom{M}{d} \geq N$.
- Let $p$ be a prime such that $p \geq \max\{m+1, |\mathcal{X}|\}$. We suppose that $\mathcal{X}$ is embedded into $\mathbb{F}_p$.
- Let $\alpha_1, \ldots, \alpha_m \in \mathbb{F}_p$ be $m$ pairwise distinct non-zero elements.
- Let $E : [N] \to \{0,1\}^M$ be an injection such that the Hamming weight of $E(x) = (E(x)_j)_{j\in[M]}$ is $d$ for any $x \in [N]$.
- Let $F_D(\boldsymbol{z}) = \sum_{x\in[N]} D_x \boldsymbol{z}^{E(x)}$ be an $M$-variate polynomial for $D = (D_1, \ldots, D_N) \in \mathbb{F}_p^N$, where $\boldsymbol{z}^{E(x)} = \prod_{j:E(x)_j=1} z_j$.

$\mathsf{Query}(x)$**.** Given an input $x \in [N]$:

1. Choose $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_t \leftarrow_{\$} \mathbb{F}_p^M$.
2. Let $\boldsymbol{q}_i = E(x) + \alpha_i \boldsymbol{v}_1 + \cdots + \alpha_i^t \boldsymbol{v}_t \in \mathbb{F}_p^M$ for $i \in [m]$.
3. Output queries $\mathsf{que}_i = \boldsymbol{q}_i$ for $i \in [m]$ and a state $\mathsf{st} = (\boldsymbol{v}_j)_{j\in[t]}$.

$\mathsf{Answer}(D, \mathsf{que}_i)$**.** Given a database $D \in \mathbb{F}_p^N$ and the $i$-th query $\mathsf{que}_i = \boldsymbol{q}_i$, output

$$\mathsf{ans}_i = \left( F_D(\boldsymbol{q}_i), \frac{\partial F_D}{\partial z_1}(\boldsymbol{q}_i), \ldots, \frac{\partial F_D}{\partial z_M}(\boldsymbol{q}_i) \right).$$

$\mathsf{Output}(\mathsf{st}, (\mathsf{ans}_i)_{i\in[m]})$**.** Given a state $\mathsf{st} = (\boldsymbol{v}_j)_{j\in[t]}$ and $m$ answers $\mathsf{ans}_i = (\zeta_i^{(0)}, \zeta_{i1}^{(1)}, \ldots, \zeta_{iM}^{(1)})$:

1. Compute

$$u_i^{(0)} = \zeta_i^{(0)},\ u_i^{(1)} = \langle \boldsymbol{\zeta}_i^{(1)}, \boldsymbol{v}_1 + 2\alpha_i \boldsymbol{v}_2 + \cdots + t\alpha_i^{t-1} \boldsymbol{v}_t \rangle$$

   for all $i \in [m]$, where $\boldsymbol{\zeta}_i^{(1)} = (\zeta_{i1}^{(1)}, \ldots, \zeta_{iM}^{(1)})$.
2. Find a degree-$dt$ polynomial $f$ such that $f(\alpha_i) = u_i^{(0)}$ and $f'(\alpha_i) = u_i^{(1)}$ for all $i \in [m]$ using Hermite interpolation.
3. Output $f(0)$.

Figure 13: Woodruff-Yekhanin's PIR [42]

**Notations.**

- Let $k \in \mathbb{N}$ be a parameter.
- Let $p$ be a prime such that $p \geq m$.
- Let $\alpha_1, \ldots, \alpha_m \in \mathbb{F}_p$ be $m$ pairwise distinct elements.

**Input.** $\boldsymbol{c} = (c_{i0}, c_{i1})_{i\in[m]} \in (\mathbb{F}_p^2)^m$.

**Output.** All degree-$k$ polynomials $f$ such that $\mathsf{dist}(\boldsymbol{c}, (f(\alpha_i), f'(\alpha_i))_{i\in[m]}) \leq b$.

**Algorithm.**

1. Find $L+1$ polynomials $R_\nu \in \mathbb{F}_p[X]$ for $\nu \in [0..L]$ such that
   - at least one of the $R_\nu$'s is not the zero polynomial;
   - $\deg R_\nu \leq 2(m-b) - \nu k - 1$ for all $\nu \in [0..L]$. If $2(m-b) - \nu k \leq 0$, we set $R_\nu(X) = 0$;
   - the $R_\nu$'s satisfy

$$\sum_{\nu=0}^{L} c_{i0}^{\nu} R_\nu(\alpha_i) = 0 \text{ and } \sum_{\nu=0}^{L} \left( c_{i0}^{\nu} R'_\nu(\alpha_i) + \nu c_{i0}^{\nu-1} c_{i1} R_\nu(\alpha_i) \right) = 0 \tag{21}$$

   for all $i \in [m]$.

   Define $R(X,Y) = \sum_{\nu=0}^{L} Y^\nu R_\nu(X)$.
2. Output all degree-$k$ polynomials $f \in \mathbb{F}_p[X]$ such that $Y - f(X)$ divides $R(X,Y)$ and $\mathsf{dist}(\boldsymbol{c}, (f(\alpha_i), f'(\alpha_i))_{i\in[m]}) \leq b$.

Figure 14: A list decoding algorithm for multiplicity codes

$\mathsf{Output}'(\mathsf{st}, (\mathsf{ans}_i)_{i\in[m]})$**.** Given a state $\mathsf{st} = (\boldsymbol{v}_j)_{j\in[t]}$ and $m$ answers $\mathsf{ans}_i = (\zeta_i^{(0)}, \zeta_{i1}^{(1)}, \ldots, \zeta_{iM}^{(1)})$:

1. Compute

$$u_i^{(0)} = \zeta_i^{(0)},\ u_i^{(1)} = \langle \boldsymbol{\zeta}_i^{(1)}, \boldsymbol{v}_1 + 2\alpha_i \boldsymbol{v}_2 + \cdots + t\alpha_i^{t-1} \boldsymbol{v}_t \rangle$$

   for all $i \in [m]$, where $\boldsymbol{\zeta}_i^{(1)} = (\zeta_{i1}^{(1)}, \ldots, \zeta_{iM}^{(1)})$.
2. Set $k = dt$ and run the algorithm in Theorem H.1 on input $\boldsymbol{c} = (u_i^{(0)}, u_i^{(1)})_{i\in[m]} \in (\mathbb{F}_p^2)^m$. Let $\mathcal{L}'$ be an obtained list of degree-$dt$ polynomials.
3. Output $\mathcal{L} = \{f(0) : f \in \mathcal{L}'\}$.

Figure 15: A list decoding algorithm for Woodruff-Yekhanin's PIR

*Proof.* Let $\Pi$ be the PIR protocol in Figure 13. Note that $F_D(\boldsymbol{q}_i) = F_D(E(x) + \alpha_i \boldsymbol{v}_1 + \cdots + \alpha_i^t \boldsymbol{v}_t)$. Define $g(X) = F_D(E(x) + X\boldsymbol{v}_1 + \cdots + X^t \boldsymbol{v}_t) \in \mathbb{F}_p[X]$. We have that $g(0) = F_D(E(x)) = D_x$. The degree of $g$ is at most $dt$ since the Hamming weight of $E(x')$ is $d$ for any $x' \in [N]$. Let

$$\boldsymbol{\zeta}_i^{(1)} = \left( \frac{\partial F_D}{\partial z_j}(\boldsymbol{q}_i) \right)_{j \in [M]} = \left( \frac{\partial F_D}{\partial z_j}(E(x) + \alpha_i \boldsymbol{v}_1 + \cdots + \alpha_i^t \boldsymbol{v}_t) \right)_{j \in [M]}.$$

If the $i$-th server is honest, it holds that

$$u_i^{(0)} = g(\alpha_i) \text{ and } u_i^{(1)} = \langle \boldsymbol{\zeta}_i^{(1)}, \boldsymbol{v}_1 + 2\alpha_i \boldsymbol{v}_2 + \cdots + t\alpha_i^{t-1} \boldsymbol{v}_t \rangle = g'(\alpha_i).$$

Note that the condition (22) is a sufficient condition for (20) with $k = dt$ since if $m \geq b + kL/2$, then $2(m-b) - \nu k \geq 0$ for all $\nu \in [0..L]$ and in this case, (20) is written as $m > b(L+1)/L + (L+1)dt/4$. In view of Theorem H.1, the algorithm shown in Figure 15 outputs the list $\mathcal{L}'$ of size at most $L$ containing all degree-$dt$ polynomials $f$ such that $\mathsf{dist}((f(\alpha_i), f'(\alpha_i))_{i \in [m]}, (u_i^{(0)}, u_i^{(1)})_{i \in [m]}) \leq b$. Therefore, $g$ is contained in $\mathcal{L}'$ and hence $D_x = g(0) \in \mathcal{L}$. Clearly, $|\mathcal{L}| \leq |\mathcal{L}'| \leq L$. Since the computation complexity of the list decoding algorithm is polynomial in $m$, $p = O(m + |\mathcal{X}|)$ and $L$, the computation complexity of $\Pi$ is polynomial in $m$, $|\mathcal{X}|$ and $L$. $\square$