Retrieve-Compute PIR and Its Applications*

Benny Applebaum
Tel-Aviv University
Tel-Aviv, Israel
benny.applebaum@gmail.com

Shahar Shechter
Tel-Aviv University
Tel-Aviv, Israel
shahar.shechter@gmail.com

August 9, 2026

Abstract

Two-server Private Information Retrieval achieves arbitrarily small polynomial communication, but relies on a strong non-collusion assumption that is difficult to justify in practice.

We introduce a new variant of two-server PIR in which one server acts as a standard *compute* server, while the other is a restricted *retrieval-only* server. The latter stores a public encoding of the database and merely serves requested symbols or blocks of this encoding, without performing any PIR-specific computation. We argue that such a retrieval-only server can be instantiated using existing static-content or repository-hosting services, thereby grounding the non-collusion assumption in realistic architectural and deployment constraints.

Assuming Learning Parity with Noise (LPN) over the ternary field with inverse-polynomial noise rate, we construct RC-PIR with arbitrarily small polynomial communication and polynomial storage. Leveraging this construction, we derive the following unexpected applications for every constant k:

1. k-server PIR with arbitrarily small polynomial communication and privacy against any coalition of $k-1$ servers.
2. k-server robust PIR with arbitrarily small polynomial communication that simultaneously achieves correctness and privacy with respect to an arbitrary monotone access structure $\mathcal{A}$. Namely, correctness is guaranteed whenever the set of online servers S satisfies $S \in \mathcal{A}$, while privacy holds against every coalition $S \notin \mathcal{A}$.
3. A k-party secure computation protocol for size-n truth tables also known as lookup tables, with arbitrarily small polynomial communication and passive security against any coalition of $k-1$ parties. This result extends to active security either in the honest-majority setting, or without an honest majority assuming collision-resistant hash functions.

None of these results were previously known under the LPN assumption. Along the way, we uncover new relationships between different complexity measures of PIR.

1 Introduction

Private Information Retrieval (PIR), introduced by Chor et al. [35], enables a client to retrieve an item from a database without revealing which item is being accessed. More formally, in a k-server t-private PIR protocol, the client sends k queries $(q_1, \ldots, q_k)$ to the k servers storing copies of the database $x \in \{0,1\}^n$, with the

*This research is supported by Israel Science Foundation (ISF) grant no. 2805/21 and by the European Union (ERC-2022-ADG) under grant agreement no.101097959 NFITSC.

guarantee that for any subset of at most t servers, the combined view of the queries they receive reveals no information about the requested index. The privacy guarantee can be either computational or information-theoretic; throughout this work, we consider computationally private PIR schemes unless stated otherwise. The traditional complexity measure for PIR is the total communication cost. We say that a family of PIR protocols has *short* communication if it can achieve an arbitrarily small polynomial communication cost (i.e., n^ϵ for arbitrary small $\epsilon > 0$).

The most widely studied and deployed PIR settings are the single-server model and the two-server model with privacy against one server, which we hereafter abbreviate as 1PIR and 2PIR, respectively. In 1PIR, short-communication protocols can be constructed under standard cryptographic assumptions, such as additively homomorphic encryption, starting with the seminal work of Kushilevitz and Ostrovsky [62]. However, these schemes generally impose significant computational overhead on the server.

In contrast, the 2PIR setting admits significantly more efficient protocols, at the cost of assuming that the two servers do not collude. In 2PIR, short communication can be achieved either information-theoretically, following the breakthrough result of Dvir and Gopi [45], or computationally under minimal symmetric-key assumptions (namely, the existence of one-way functions), as first shown by Chor and Gilboa [34] and later substantially improved by Gilboa et al. [47, 26]. The latter class of schemes is particularly efficient and practical. Nevertheless, in many real-world scenarios, the non-collusion assumption can be difficult to justify.

To illustrate this point, suppose that a data owner, Alice, wishes to grant clients private access to her data using a two-server PIR scheme. She may rent two servers, A_1 and A_2, from cloud providers and deploy a two-server PIR protocol on them. However, from the client's perspective, the non-collusion requirement relies on several additional assumptions:

- The server implementations are correct and do not contain "hidden" functionality that enables the servers to share information.
- The data owner Alice has sufficiently limited access to the servers; otherwise, she could log internal states, monitor communication, or collect auxiliary information that may allow her to infer the client's queried index.

These assumptions can be difficult to satisfy in practice. More generally, it appears conceptually problematic, or even paradoxical, to assume that a single untrusted entity (Alice) can reliably set up and maintain a pair of non-colluding entities.[1]

This issue can be addressed by resorting to trusted hardware or by using 1PIR. In this paper, we propose an alternative, lightweight solution that also leads to additional, perhaps surprising, applications.

1.1 Retrieve-Compute PIR

We introduce *Retrieve-Compute PIR* (RC-PIR), a restricted two-server PIR model in which one server is limited to retrieving data from a *publicly* encoded version of the database. Specifically, we consider two servers, denoted S_C and S_R, where S_C is a compute server and S_R is a restricted retrieval server. During a setup phase, the database x is mapped via a public encoding into an encoded database R, which is stored on S_R.

A client wishing to access an item $i \in [n]$ generates a pair of queries (q_C, q_R), which are sent to S_C and S_R, respectively. The compute server responds, as usual, by applying a computation $\textsc{CAnswer}(x, q_C)$,

[1] This concern does not arise in scenarios that inherently involve multiple independent parties, such as when PIR is deployed within a secure multiparty computation framework.

while the retrieval server interprets q_R as a list of addresses (i.e., $q_R \subset [|R|]$) and returns the corresponding entries $(R_j : j \in q_R)$ of the encoded database R. Importantly, the privacy definition remains unchanged: for any efficient adversary, observing either q_C or q_R reveals no information about the queried index i.

The main motivation for this model is straightforward: by restricting the functionality required of one of the servers, we aim to make the non-collusion assumption significantly easier to realize in practice. Returning to the above example, RC-PIR allows Alice to implement the compute server locally, while delegating the retrieval server to any third-party service that supports simple "retrieve-only" functionality, namely, the ability to expose files over the internet without performing additional computation. Examples of suitable retrieval services include:

- **Code repositories (e.g., GitHub, Bitbucket):** the repository owner cannot learn which specific files are accessed.
- **Decentralized hosting:** services such as BitTorrent or Freenet, where files are hosted by many independent peers; as the number of peers increases, it becomes increasingly unlikely that any of them is associated with the compute server.
- **Cloud object storage (e.g., AWS S3):** although cloud providers typically support access logging, it is possible to verify that logging is disabled by creating a public user with permissions to check the logging configuration. Similar mechanisms appear to be available in other cloud platforms, such as Google Cloud, making it feasible to use a single provider for both servers while maintaining separation of roles.

In all these examples, the retrieval server is limited to servicing file-access requests via a fixed interface and neither observes the full client query nor has any channel to coordinate with the compute server, which makes the non-collusion assumption substantially easier to justify. Overall, RC-PIR provides a compelling two-server PIR model in which non-collusion follows from architectural and deployment constraints, rather than from assuming trust between independent entities.

1.2 Constructing RC-PIR based on LPN

Our main result is a construction of RC-PIR with short communication and almost-linear storage. The construction is based on the Learning Parity with Noise (LPN) assumption [19, 50] over the ternary field $\mathbb{F}_3$, with constant rate and arbitrary inverse-polynomial noise rate. We refer to this assumption hereafter simply as LPN. The binary low-noise version of this assumption has been widely studied and used in cryptographic constructions, even with more aggressive parameters [4, 44, 61, 43, 1], as have its extensions to other finite fields [56, 6, 20, 21, 58, 59, 31]. We assume only standard polynomial hardness, though the best known attacks are sub-exponential.

Informal Theorem 1.1 (main)**.** *Assuming LPN, for every constant $\epsilon > 0$, there exists a two-server RC-PIR protocol with communication complexity $O(n^\epsilon)$ and polynomial storage complexity $O(n^{c_\epsilon})$, where n is the length of the database and the constant c_ϵ increases as ϵ decreases.*

We make the following remarks.

- The LPN assumption is not known to imply 1PIR with short communication.[2] Thus, the theorem shows that short RC-PIR can be based on assumptions that appear weaker than those currently known to yield short 1PIR, yet stronger than those sufficient for short 2PIR.

[2]The best known construction, due to Abram et al. [1], achieves only slightly sublinear communication, namely $n/2^{\log^{1-\beta} n}$ for some constant $\beta > 0$, based on LPN in the so-called low-noise regime $O(\log^{1+\beta} \lambda/\lambda)$.

- Viewed abstractly, short RC-PIR lies between these two primitives and can be informally interpreted as PIR with "1.5 servers." Indeed, we have the trivial implications

$$\text{short 1PIR} \implies \text{short RC-PIR} \implies \text{short 2PIR}.$$

  A more detailed comparison with other PIR models is deferred to Section 1.6.

- The storage exponent deteriorates rapidly as ϵ decreases; specifically,

$$c_\epsilon = \epsilon \exp\big(O(\epsilon^{-1}\log(\epsilon^{-1}))\big).$$

  Consequently, Theorem 1.1 should primarily be viewed as a theoretical feasibility result. Nevertheless, we believe that the underlying techniques can be adapted to yield more concretely efficient instantiations. See Remark 2.1 for a preliminary step in this direction.

Although the notion of RC-PIR is motivated by practical trust considerations, it also gives rise to several interesting applications to classical problems in the theory of PIR and secure multiparty computation (MPC). We conclude this section with a brief overview of these applications.

1.3 Multi-server PIR with full privacy and short communication

Let us now move beyond 1PIR and 2PIR to the more general setting of k-server PIR. In this regime, most existing protocols guarantee privacy only against a single server (i.e., 1-privacy) and become completely insecure in the presence of two colluding servers. Although there exist PIR protocols with a larger privacy threshold $t > 1$, both in the information-theoretic setting [12, 14, 54, 71, 9] and in the computational setting [24, 26, 27, 48], their communication cost is typically significantly higher.

In particular, in the case of *full privacy*, namely $t = k - 1$, it remains unknown how to achieve communication below $O(\sqrt{n})$ information-theoretically, or below $O(n^{1/3})$ under symmetric-key assumptions.[3] Breaking these barriers in this regime has been posed as a central open problem by several researchers (see, e.g., [37]). Until recently, it was unclear whether such constructions could be obtained without relying on assumptions that imply 1PIR.

This situation changed with the recent works of Dao et al. [42] and Couteau et al. [40], who constructed such fully private multi-server PIR protocols from two different sets of assumptions. Dao et al. [42] rely on a sparse-LPN assumption, in which the LPN samples are sparse vectors over a large field and are perturbed by inverse-polynomial noise. Couteau et al. [40], on the other hand, rely on the standard (non-sparse) LPN assumption over a large field together with an additional Multivariate Quadratic (MQ) assumption. The latter posits that a random quadratic system consisting of $n^{1+\epsilon}$ equations in n variables over a sufficiently large field is pseudorandom.

A third route, pointed out to us by Yuval Ishai, becomes available if one relaxes the PIR model by allowing interactive preprocessing and a constant number of online rounds. In this setting, an LPN-based fully private k-server PIR can be obtained by distributing the client in the secret-key PIR of [31] among the servers using general MPC techniques.

Using Theorem 1.1, we eliminate the need for sparsity, the MQ assumption, and additional interaction, obtaining fully private, standard single-round PIR protocols based solely on LPN.

Informal Theorem 1.2 (fully-private PIR)**.** *Assuming LPN, for every constant $\epsilon > 0$ and constant $k \geq 2$, there exists a k-server PIR with $(k-1)$-privacy and communication complexity of $O(n^\epsilon)$.*

[3] A cubic-root construction under one-way functions follows by combining the distributed-point functions of [48, 24] with standard balancing techniques [35]. See also [23, Thm. 3.5].

Proof idea. The proof proceeds via a simple and general reduction from fully private k-server PIR to the RC model. Since the retrieval server is restricted to fetching entries from the memory, its role can be securely distributed among two additional servers using an independent instance of a two-server PIR protocol. Moreover, when the resulting protocol is implemented via RC-PIR, this transformation can be applied recursively for a constant number of levels. Crucially, we rely on the fact that the database encoding is public and privacy holds even if the "retrieve" server knows it. (See Section 5.1 for full details.)

1.4 Short PIR for arbitrary access structures

Multiple servers are also useful for introducing redundancy and coping with non-responding servers. In particular, a PIR protocol is said to be v-*correct* if correctness is guaranteed as long as at least v out of the k servers are "alive." This notion of robustness was introduced by Beimel and Stahl [16], who constructed k-server information-theoretic PIR schemes achieving t-privacy and v-correctness for various constant parameters $t < v < k$, with communication complexity $O\big(n^{1/(\lfloor (v-1)/t \rfloor + 1)}\big)$.[4] In the threshold case $v = t+1$, this yields square-root communication. To the best of our knowledge, this square-root bound remains the best known even for computational schemes, unless one assumes the existence of 1PIR.

We show that this bound can be improved under the LPN assumption. In fact, we handle the more general setting of an arbitrary constant-size *access structure*. Specifically, given any monotone function $f : 2^{[k]} \to \{0,1\}$, we construct a PIR protocol that guarantees correctness in the presence of any subset of servers $S \subseteq [k]$ satisfying $f(S) = 1$, and guarantees privacy against any coalition $S \subseteq [k]$ for which $f(S) = 0$. (Following Footnote 4, we assume that every subset of servers provides either privacy or correctness, as requiring both would immediately imply 1PIR.)

Informal Theorem 1.3 (PIR with general access structure)**.** *Assuming LPN, for every constant $\epsilon > 0$, constant $k \geq 2$ and monotone function $f : 2^{[k]} \to \{0,1\}$, there exists a k-server PIR that realizes f with communication complexity of $O(n^{\epsilon})$.*

To the best of our knowledge, this is the first nontrivial PIR construction for general access structures that does not rely on 1PIR. The proof combines Theorem 1.2 with a simple replication-based construction [16].

As in formula-based secret sharing [18], we represent the access structure f as a monotone CNF or DNF formula and realize it recursively: unbounded AND gates are implemented via Theorem 1.2, while unbounded OR gates are handled by replication. Observe that we cannot recurse twice over AND gates, and hence the construction is limited to formulas of AND-depth 1. (In fact, we can handle slightly more powerful formulas; see Remark 5.5.) Fortunately, such formulas are expressive enough to realize every monotone function. (See Section 5.2 for full details.)

We further note that, in the case of a threshold access structure with $t < k/3$, one can additionally guarantee correctness even in the presence of t malicious (i.e., Byzantine [16]) servers. Indeed, by adapting analogous transformations from the secret-sharing literature (see, e.g., [41]), the reduction extends more generally to any (Q3) access structure, namely one in which no three unauthorized sets cover all parties. (See Section 5.2 for full details.)

[4] It is not hard to see that a scheme with $t = v$ implies a single-server PIR. (Just send the first t queries to the single server and let her compute the responses.) Therefore, the restriction $t < v$ is essentially the strongest guarantee one can hope for without resorting to 1PIR.

1.5 General MPC with short communication for truth-tables

PIR protocols can be viewed as a special case of MPC protocols, and there are well-known connections between low-communication PIR protocols and low-communication MPC protocols whose communication is smaller than the representation size of the computed function [55, 13]. Indeed, the construction underlying Theorem 1.2 gives rise to such an MPC protocol.

Informal Theorem 1.4 (MPC with arbitrarily small polynomial communication)**.** *Assuming LPN, the following holds. For every constant $\epsilon > 0$, every $k \geq 2$, and every finite k-party functionality f represented by an n-size truth table, there exists a k-party MPC protocol with $\mathrm{poly}(n)$ efficiency and communication complexity of $O_k(n^\epsilon)$, secure against passive $\mathrm{poly}(n)$-size adversaries corrupting an arbitrary subset of the parties. Moreover, there are protocols with similar efficiency that achieve security against active $\mathrm{poly}(n)$-size adversaries: with guaranteed output delivery assuming an honest majority, and with abort against arbitrary corruptions assuming collision-resistant hash functions.*

We note that MPC protocols for truth-tables (aka lookup-tables) form a useful building block both in theory and in practice, see the discussion in [3].

Proof idea. To prove the theorem, we observe that the PIR protocol from Theorem 1.2 is not only communication-efficient but also computationally efficient: both the query-generation algorithm and the decoding algorithm can be implemented by circuits of size $\tilde{O}(n^\epsilon)$. We can therefore securely compute $f(y_1, \ldots, y_k)$ as follows. View the truth table of f as a database F, and interpret the joint input $y = (y_1, \ldots, y_k)$ as an index into F. The protocol proceeds in three steps:

1. The parties run a generic MPC protocol for the functionality g that, on input $y = (y_1, \ldots, y_k)$ and randomness $(r_1, \ldots, r_k)$, computes the PIR queries $q_1, \ldots, q_k$ with respect to an index y and random tape $r = \sum_i r_i$, and delivers q_i to party i.

2. Each party locally computes its PIR answer a_i based on F and q_i.

3. The parties jointly run an MPC protocol to evaluate the decoding functionality D on their answers (and on their input/randomness pair (y_i, r_i)).

Steps (1) and (3) can be realized using standard MPC protocols (e.g., [51]), with communication proportional to the circuit size of g and D, namely $O_k(n^\epsilon)$.[5] We further show how to extend this template to the setting of an active adversary. For this one has to make sure that the calls to g and D and the local computation are all consistent. This is relatively straightforward if one assumes succinct zero-knowledge arguments (whose existence follows from collision-resistant hash functions [**?**]). Interestingly, in the honest majority setting, this can also be done without any zero-knowledge machinery.

As already mentioned, the connection between low-communication PIR and low-communication MPC has been observed previously in different MPC settings, and we thank the authors of [13] for bringing it to our attention. Since we could not find a formal statement in the literature relating short-communication fully private PIR to low-communication MPC—or even a description of the corresponding reduction—we present the corresponding reductions in full generality and detail in Section 6. Moreover, the transformation yielding actively secure protocols in the honest-majority setting without relying on succinct proofs appears to be new.

[5] For this, we require an Oblivious Transfer (OT) protocol whose existence follows from the LPN assumption [4].

Previous protocols for sub-linear MPC. MPC protocols with sub-linear communication have been extensively studied in recent years. While such protocols can be realized under fully homomorphic encryption, a recent line of work has shown how to obtain sub-linear communication under progressively weaker assumptions.

Theorem 1.4 continues this line of research by providing the first general multi-party protocol with sub-linear communication based solely on the standard LPN assumption. Previous works relied on stronger assumptions, including group-based assumptions [25, 2], the sparse-LPN assumption [42] (i.e., LPN with a sparse generating matrix), or combinations of LPN with additional primitives, such as constant-depth or constant-degree pseudorandom generators with large stretch (e.g., via the MQ assumption) [33, 22, 39, 40]. In some cases, further auxiliary assumptions were required. To the best of our knowledge, prior to this work, even for a restricted number of parties, the existence of a sub-linear MPC protocol based solely on the standard LPN assumption remained open.

1.6 Other related works

Online-efficient PIR $\Longrightarrow$ RC-PIR. The RC model is closely related to the notion of *online-efficient PIR* introduced by Beimel et al. [15]. In that model, all k servers (in our setting $k = 2$) may preprocess the database x via a public preprocessing function and store a polynomially larger encoding of x. In the online phase, the servers may perform arbitrary computation to produce their answers, but their *online work* — measured as the number of bits accessed in the encoded database — must be sublinear, namely $w = o(n)$. Communication complexity is treated there as a separate complexity measure. Assuming that the memory access pattern is fully determined by the queries (as is the case in most known constructions), a protocol with online work w can be transformed into an RC-PIR protocol with communication complexity $O(w \log n)$. Indeed, the client can request the w accessed entries directly from the servers and then perform the prescribed computation locally on the retrieved data.[6]

For the 2PIR setting, the best known construction of [15] achieves nearly square-root online work, namely $O(n^{0.5+\epsilon})$ for any constant $\epsilon > 0$ (see also [46, 64] for more recent related results). This construction provides information-theoretic privacy. In the computational setting, a breakthrough result of [65] gives a 1PIR with polylogarithmic online work based on the Ring-LWE assumption [67]. While this demonstrates important feasibility, the resulting protocol remains impractical. Currently, it is still unknown how to surpass the square-root online-work barrier in the two-server setting under weaker cryptographic assumptions that do not imply 1PIR with short communication, let alone unconditionally in an information-theoretic sense.

Other PIR models with private preprocessing. It is important to distinguish between our model in which the preprocessing step is *public* to other models in which the client holds a secret state that either depends on the database [38] or depends on the encoded database in a secret way [28, 29, 32]. Indeed, these models are incomparable to our model and we do not know how to translate results from our model to theirs and vice versa.

1.7 Discussion and Open Questions

RC-PIR as a bridge. Our work bridges two seemingly unrelated research directions: optimizing efficiency and strengthening security. In particular, our work reveals a tight connection between two central

[6] This transformation does not necessarily preserve the communication complexity of the original protocol, which may be smaller than w. Moreover, from our perspective this approach is somewhat excessive, as it effectively yields two retrieval servers, leaving the compute server underutilized.

barriers in the study of PIR: the square-root barrier for online work in 2PIR and the square-root barrier for communication in $(2, 3)$-PIR.

Indeed, the transformation underlying Theorem 1.2 shows that RC-PIR with sub-square-root communication n^c for $c < 0.5$ yields a 2-private 3-server PIR (i.e., $(2, 3)$-PIR) with sub-square-root communication $n^{c'}$ for $c' < 0.5$ (for example, by replacing the retrieve server with the two-server construction of [45]). Combined with the trivial implication that a 2PIR protocol with online work w yields an RC-PIR protocol with communication $\tilde{O}(w)$, we conclude that 2PIR with sub-square-root online work n^c for $c < 0.5$ implies $(2, 3)$-PIR with sub-square-root communication $n^{c'}$ for $c' < 0.5$. Thus, breaking the square-root barrier for online work would immediately break the square-root barrier for $(2, 3)$-PIR.

Importantly, these reductions are efficient and information-theoretic. Hence, this relation holds unconditionally, independent of any additional computational assumptions (e.g., one-way functions).

Which assumptions suffice for RC-PIR? Our results show that short RC-PIR can be obtained from assumptions that appear weaker than those known to imply short 1PIR. This naturally raises the question of whether the two notions can be formally separated. More ambitiously, can short RC-PIR be based on symmetric primitives such as one-way functions, random oracles, or LPN with constant noise rate?

Establishing lower bounds in this direction appears challenging. Indeed, 2PIR with logarithmic online work (which implies RC-PIR with logarithmic communication) can be obtained from any information-theoretic 2PIR with logarithmic communication via [15]. The existence of such information-theoretic constructions is a long-standing open problem. Therefore, ruling out short RC-PIR—even under very weak assumptions—would require resolving this central open question in the negative.

A possible workaround would be to prove such a negative result under the assumption that no information-theoretic 2PIR with logarithmic communication exists, that is, that symmetric-key assumptions do not provide additional power in this regime. (See, e.g., [7, 66] for results in a similar spirit.) We leave this intriguing direction for future work.

Organization. Section 2 provides an overview of our techniques. Following the preliminaries in Section 3, we present our RC-PIR constructions in Section 4. We then describe several PIR applications in Section 5, including fully private (k)-server PIR, PIR for general access structures, and PIR with Byzantine servers. Finally, Section 6 discusses applications to MPC. Some proofs are deferred to the appendices.

2 Technical Overview

We present an overview of the RC-PIR construction. To illustrate the main idea, we begin with a toy example that combines the Hadamard-based PIR with (binary) LPN. Although this construction incurs relatively high communication cost, it captures the structural framework underlying our subsequent constructions.

2.1 RC-PIR from Hadamard and LPN

In the Hadamard-based 2PIR scheme [35], the database $x \in \mathbb{F}_2^n$ is viewed as a binary vector. A client holding an index i wishes to retrieve x_i, namely the mod-2 inner product $\langle x, 1_i \rangle$, where 1_i denotes the i-th standard basis vector. To this end, the client secret-shares 1_i into a pair of queries $q_1 \xleftarrow{R} \mathbb{F}_2^n$ and $q_2 = q_1 + 1_i$, sends q_1 and q_2 to the first and second servers, respectively, and receives the answers $a_1 = \langle x, q_1 \rangle$ and $a_2 = \langle x, q_2 \rangle$. The client then computes

$$a_1 + a_2 = \langle x, q_2 - q_1 \rangle = x_i.$$

The protocol achieves perfect privacy, since the marginal distribution of each query is independent of the index i.

Before moving to the RC-PIR setting, we slightly modify the protocol by replacing q_1 with a pseudorandom vector sampled from the LPN distribution. It is convenient to work with the computationally equivalent dual formulation of LPN. Let H be a public $n \times 2n$ parity-check matrix, and let $e \in \mathbb{F}_2^{2n}$ be a random sparse vector of Hamming weight at most $s = n^\epsilon$. We define $q_1 = He$ (i.e., a random syndrome), and set $q_2 = q_1 + 1_i$ as before. The protocol remains correct; privacy against the first server is still information-theoretic, while privacy against the second server holds computationally under the LPN assumption.

To obtain an RC-PIR protocol, observe that the first server computes $\langle x, He \rangle$, which can be decomposed into an offline encoding $R = x^T H \in \mathbb{F}_2^{2n}$ and an online computation $\langle R, e \rangle$. Moreover, the noise vector e can be safely sent to the server, as it is sampled independently of the index i (and thus preserves privacy against the second server). Finally, since e is s-sparse, computing $\langle R, e \rangle$ reduces to fetching s entries from the encoded database. This yields an RC-PIR protocol in which the compute server holds x, the retrieval server holds the encoded database $R = x^T H$, and the retrieval phase reduces to fetching the s positions indexed by the support of e. (See Section A for a detailed description of this toy example and its optimized versions.)

2.2 RC-PIR from low-degree 2PIR and LPN

While the Hadamard-based construction can be mildly improved via standard balancing techniques, the resulting protocol still incurs square-root communication. Nevertheless, we can attempt to extend the above approach to more general PIR protocols. To this end, let us abstract the key properties of the Hadamard construction that were used:

1. For every fixed database x, the servers compute an $\mathbb{F}_2$-linear function A_x of the query.
2. The marginal distribution of the first query is uniform, and the second query can be derived from q_1 and the index i via an efficient computation.

A closer look reveals that this template extends naturally to the case where $A_x : \mathbb{F}^t \to \mathbb{F}^\ell$ is a low-degree polynomial mapping over some field $\mathbb{F}$, mapping a query $q \in \mathbb{F}^t$ to an answer $a \in \mathbb{F}^\ell$.

Concretely, suppose that $A_x(\cdot)$ has degree d. Let H be a random $t \times m$ matrix over $\mathbb{F}$ with $m = 2t$, and define the derived function $A'_{x,H}$ that, on input a syndrome $e \in \mathbb{F}^m$, outputs $A_x(He)$. Since H is linear over $\mathbb{F}$, the mapping A' also has degree d.

Moreover, if e is a sparse vector, we can encode A' as a vector R such that $A'(e)$ can be evaluated by accessing only a small number of entries of R. Specifically, if R is taken to be the coefficient vector of A', then computing $A'(e)$ requires accessing at most $O(s^d)$ entries per output coordinate, where s denotes the Hamming weight of e. Indeed, writing each output coordinate of A' as a degree-d polynomial in $(e_1, \ldots, e_m)$, observe that only monomials fully supported on $\mathrm{supp}(e)$ can contribute to the evaluation. As $|\mathrm{supp}(e)| = s$ and d is constant, there are only $O(s^d)$ such monomials, and hence only $O(s^d)$ coefficients need to be accessed.

This yields an RC-PIR protocol in which the upload and download complexity of the compute server remain unchanged, namely t and ℓ, respectively. The upload and download complexity of the retrieval server are $O(\ell s^d)$,[7] and the storage complexity is $O(\ell m^d) = O(\ell (2t)^d)$ field elements.

[7] If we view the database as consisting of ℓ blocks (one per output coordinate of A_x), then the client need only transmit $O(s^d)$ block addresses. In this case, the upload complexity is $O(s^d)$ and the download complexity is $O(\ell s^d)$.

Remark 2.1 (A cube-root construction)**.** *We note that the cube-root derivative-based construction of [71] satisfies the above properties, yielding an RC-PIR scheme with nearly cube-root communication; see Corollary 4.3. Although this does not suffice to establish Theorem 1.1, the resulting construction may still offer favorable concrete efficiency. In ongoing work, we show that by optimizing this approach one can obtain a practical scheme with linear storage and cube-root communication.*

2.3 Low-degree PIR with low communication?

Short-communication 2PIR protocols are known, either computationally via distributed point functions [47, 26] or information-theoretically via matching vectors (MV) [45]. It turns out that the former does not satisfy the second property of our template above, as it is unclear how to derive the second query from a uniformly chosen first query in an efficient manner. We therefore focus on the latter approach.

To obtain a low-degree 2PIR, we consider the recent protocols of Alon et al. and Beimel et al. [11, 5], which are based on a *sparse* variant of matching vectors. Roughly speaking, their protocol has a subtle structure that combines mod-2 and mod-3 computations[8]:

1. Let d be a constant sparsity parameter, and let h be a communication parameter that can be set to n^{ϵ_d}, where ϵ_d decreases as d increases.

2. To generate the queries $q_1, q_2 \in \mathbb{Z}_2^h$ we sample a $\mathbb{Z}_2$-additive secret sharing of a d-sparse vector $u = u(i)$ determined by the index i.

3. Given a query q, each server computes a mapping $B_x(q) = (b_1, \ldots, b_n)$, where B_x is a d-local linear function of q over $\mathbb{Z}_2$. (Here B stands for "binary.")

4. Each server then *casts* the values $b_1, \ldots, b_n$ as $0/1$ elements over $\mathbb{Z}_3$, computes some linear operation over $\mathbb{Z}_3$, and returns the result. Denote this ternary aggregation by $T : \{0,1\} \to \mathbb{Z}_3^h$. (Here T stands for "ternary.")

(We omit the client's decoding procedure, which is not relevant here.)

The server computation thus consists of two linear mappings over different domains. However, their composition is not linear over either domain.[9]

The key observation is that the binary component B is not only low-degree but also low-locality: each output depends on at most d entries of q. Therefore, if we interpret q as a $0/1$ vector over $\mathbb{Z}_3$, we can rewrite B as a polynomial mapping over $\mathbb{Z}_3$ of degree $O(d)$. Consequently, the overall server computation $A_x(q)$ can be expressed as a low-degree mapping over $\mathbb{Z}_3$, provided that the inputs q_1, q_2 are restricted to $\{0,1\}$ entries. Correctness is preserved as long as q_1 and q_2 form a secret sharing of u modulo 2.

This raises the question over which domain the LPN pseudorandom mask should be generated. If we work over $\mathbb{Z}_2$, then the preprocessed answer function $A'_x(e) = A_x(He)$ is no longer low-degree, as it becomes a composition of a non-local mod-2 linear mapping H and a low-degree mod-3 function A_x.

On the other hand, using LPN over $\mathbb{Z}_3$ preserves low degree under composition, but introduces two new challenges. First, the query q_1 may now take ternary values, whereas the computation assumes $0/1$ inputs. This can be addressed by having the first server apply a casting operator that maps 2 to 0 while leaving 0 and 1 unchanged; this operation can be implemented by a degree-2 polynomial over $\mathbb{Z}_3$.

[8] This approach can be generalized to any pair of primes.

[9] Nor can we embed the computation as a low-degree polynomial over $\mathbb{Z}_6$. To see this, recall that, by the Chinese Remainder Theorem, any polynomial mapping over $\mathbb{Z}_6$ decomposes into independent mappings over $\mathbb{Z}_2$ and $\mathbb{Z}_3$, whereas here the casting operator couples the two components.

The second issue concerns the query q_2. To computationally hide the vector $u = u(i)$, we mask it as $q_2 = He + u \pmod 3$, relying on the pseudorandomness of He over $\mathbb{Z}_3$. However, q_1 and q_2 may no longer form a secret sharing of u over $\mathbb{Z}_2$. Fortunately, sparsity again helps: since u is d-sparse (for constant d), the pair (q_1, q_2) forms a $\mathbb{Z}_2$-additive secret sharing of u with constant probability $(2/3)^d$. Since the client knows when failure occurs, this success probability can be easily amplified (e.g., via repetition) at only a minor, say polylogarithmic, increase in communication.

Applying the RC transformation to this low-degree PIR construction completes the proof of Theorem 1.1.

Remark 2.2 (Abstraction: data-structure perspective)**.** *It is instructive to view the construction through the following data-structure lens. Associate with a PIR scheme the family of answer functions $\mathcal{F} = \{f_x\}_x$, where $f_x(q)$ denotes the answer produced by a server holding database x on query q. To obtain an RC-PIR scheme, it suffices to represent these answer functions by a cell-probe data structure: namely, to encode x into a database R such that $f_x(q)$ can be recovered from a small number of probes to R. In this interpretation, the retrieval server stores R and returns the probed cells, while the client performs the final local reconstruction. Thus, for privacy, it suffices that the probe pattern seen by the retrieval server be determined only by q and public parameters; the retrieval server's view then hides the index by post-processing.*

The LPN assumption allows us to reduce the original family $\mathcal{F}$ to the derived family $\mathcal{F}' = \{g_x\}_x$, where

$$g_x(e) = f_x(He),$$

for a public matrix H. Thus, instead of building a data structure for $f_x(q)$ on arbitrary PIR queries q, it suffices to build one for $g_x(e) = f_x(He)$ on sparse inputs e. This restriction is what enables the succinct cell-probe representation used in our construction.

3 Preliminaries

General Notation. For an integer $n \in \mathbb{N}$ and subset $S \subseteq [n]$, let $1_S \in \{0,1\}^n$ denote the characteristic vector of S, which has 1 at the j-th coordinate if $j \in S$ and 0 otherwise. For a single index $i \in [n]$, we write 1_i to denote the vector $1_{\{i\}}$. For a function (or algorithm) A, we let $S(A)$ denote the circuit size of A.

3.1 Private Information Retrieval

Definition 3.1 (k-server PIR)**.** *A k-server PIR scheme defined by the tuple of algorithms* (QUERY, ANS, DEC)*. Let $x \in \{0,1\}^n$ be a database of size n.*

- *The* $\text{QUERY}(1^n, i; r)$ *algorithm that given an index $i \in [n]$ and randomness outputs a set of queries $(q_1, q_2, \ldots, q_k)$. For $s \in [k]$, we write* $\text{QUERY}_s(1^n, i; r)$ *to denote the sth output of the algorithm, i.e., q_s. We typically omit the input 1^n when n is clear from the context.*
- *The* $\text{ANS}(s, x, q)$ *algorithm that given server s, the database x and a query q_s, outputs an answer $\mathbf{ans}_s$.*
- *The* $\text{DEC}(1^n, i, r, \mathbf{ans}_1, \mathbf{ans}_2, \ldots, \mathbf{ans}_k)$ *that given the answers, randomness and index $i \in [n]$ outputs a single bit $y \in \{0,1\}$. We typically omit the input 1^n when n is clear from the context.*

The algorithms should satisfy the following properties:

- *δ-Correctness: For every database $x \in \{0,1\}^n$ and every index $i \in [n]$, consider the random variables*

$$(q_1, q_2, \ldots, q_k) = \text{QUERY}(i; r) \quad \textit{and} \quad \mathbf{ans}_j = \text{ANS}(j, x, q_j), \forall j \in [k], \tag{1}$$

that are induced by a random choice of the coins r of the client, then the error probability

$$\Pr_r[\text{DEC}(i, r, (\mathbf{ans}_1, \ldots, \mathbf{ans}_k)) \neq x_i]$$

is at most δ. When δ is not specified, we assume, by default, that it is negligible in n.

- *t-Privacy: For every coalition $T \subseteq [k]$ of size at most t and every sequence of pairs of indices $(i_n, j_n)_{n\in\mathbb{N}}$ where $i_n, j_n \in [n]$, the following distributions are computationally indistinguishable*

$$(\text{QUERY}(1^n, i; r))_T \approx_c (\text{QUERY}(1^n, j; r))_T,$$

where the ensembles are parameterized by the database length n. We say that the PIR scheme is information theoretically secure if the two distributions are identical (rather than computationally indistinguishable). Furthermore, for $k \geq 2$ we say the scheme is fully private if $t = k - 1$ servers.

The ith query complexity *(resp.,* answer complexity*) is the bit length of the ith query (resp., answer). The* upload*/*download *complexities are the the sum of the query/answer complexities over all servers. We measure all these quantities as functions of the database size n.*

Having established the standard PIR model, we introduce the Retrieve-Compute PIR (RC-PIR) variant. This is a special case of k-server PIR in which the kth server (aka the *Retrieval server*) is a restricted entity that supports only "fetch" operations. This server holds a pre-processed database defined as a set of entries which may be requested by address. The following definition assumes that the encoded database is partitioned into blocks and that the client always asks for an entire block. This choice is aligned with our constructions. Still when analyzing communication we will count the number of bits. This means that if the encoded database has B blocks of size ℓ-bits each, and the client fetches F addresses the query/answer complexity of the retrieval server are $F \log B$ and $F\ell$, respectively.

Definition 3.2 (k-server RC-PIR)**.** *A k-server PIR protocol* $(\text{QUERY}, \text{ANS}, \text{DEC})$ *is an RC-PIR protocol if there exists a (possibly randomized) encoder algorithm* ENCODER *such that the following hold:*

1. *The algorithm* ENCODER *maps the database $x \in \{0,1\}^n$ to an encoding $R = (R_1, \ldots, R_B)$ with $B = B(n)$ entries each of bit-length $\ell = \ell(n)$. We refer to ℓ as the* block size *and to B as the* number of blocks *and to $N = B\ell$ as the* storage complexity.

2. *The kth query $q_k = \text{QUERY}_k(1^n, i; r)$ is a sequence of indices (addresses) $(j_1, \ldots, j_F)$ in $[B]$.*

3. *The kth answer algorithm $\text{ANS}(k, x, q_k)$, parses q_k as a sequence of indices $(j_1, \ldots, j_F)$ in $[B]$ and returns $(R_{j_1}, \ldots, R_{j_F})$ where $R = \text{ENCODER}(x)$.*

Importantly, if the encoder algorithm is randomized we assume that this randomness is chosen once and for all, and is given as public randomness to all other algorithms and to the adversary.

We sometimes refer to $\text{QUERY}_k(\cdot)$ and to $\text{ANS}(k, \cdot)$ as the algorithms $\text{RQUERY}(\cdot)$ and $\text{RANSWER}(\cdot)$ to highlight that these are the query and answer algorithms of the Retrieval server. To avoid redundancy we let CQUERY *denote the query algorithm restricted to the first $k - 1$ queries that are sent to the computation servers.*

3.2 Matching Vectors Family

Definition 3.3 (Matching vector family [72]). *Let m, h, n be positive integers, and let $S, T \subseteq \mathbb{Z}_m$ be disjoint subset. The collection of vector pairs $((u_i, v_i))_{i=1}^n$ where $u_i, v_i \in \mathbb{Z}_m^h$ is an (S, T)-matching vector family if the following conditions are satisfied:*

1. $\langle u_i, v_i \rangle \mod m \in T$ *for* $i \in [n]$.
2. $\langle u_i, v_i \rangle \mod m \in S$ *for* $i \neq j \in [n]$.

Let $m = p_1 p_2$ for distinct primes $p_1 < p_2$. We categorize specific families based on the structure of the set S. In these cases, we often omit T and refer to the collection simply as an S-matching vector family:

- $S_{\mathsf{can}} = \{a \in \mathbb{Z}_m \mid (a \mod p_1 \in \{0,1\}) \wedge (a \mod p_2 \in \{0,1\})\} \setminus \{0\}$ *with* $T = \{0\}$.
- $S_{\mathsf{one}} = \{a \in \mathbb{Z}_m \mid (a \equiv 1 \pmod{p_1}) \vee (a \equiv 1 \pmod{p_2})\}$ *with* $T = \{0\}$.
- $S_{\mathsf{zero}} = \{a \in \mathbb{Z}_m \mid a \equiv 0 \pmod{p_1} \vee a \equiv 0 \pmod{p_2}\}$ *with* $T = \{1\}$.

Note that $S_{\mathsf{can}} \subseteq S_{\mathsf{one}}$ for all distinct primes p_1, p_2.

Definition 3.4 ((d, p)-Sparse Matching Vectors [11]). *Let $((u_i, v_i))_{i=1}^n$ be an (S, T)-matching vector family over $\mathbb{Z}_m^h$ for some $m, h \in \mathbb{N}$. We say that $((u_i, v_i))_{i=1}^n$ is a (d, p)-sparse (S, T)-matching vector family if for all $i \in [n]$,*

$$|\{\ell \in [h] \mid v_i[\ell] \not\equiv 0 \pmod{p}\}| \leq d \quad \textit{and} \quad |\{\ell \in [h] \mid u_i[\ell] \not\equiv 0 \pmod{p}\}| \leq d.$$

Theorem 3.5 ([11]). *For every $n, d > 0$, there is a $(d, 2)$-sparse S_{can}-matching vector family $((u_i, v_i))_{i \in [n]}$ over $\mathbb{Z}_6^h$ with $h \leq d^{O(1)} n^{O(\frac{\log \log d}{\log d})}$.*

Claim 3.6. *Let d be a positive integer, p_1, p_2 be distinct primes and let $m = p_1 p_2$. If there is a (d, p_1)-sparse S_{one}-matching vector family over $\mathbb{Z}_m$ of length h, then there is a $(d+1, p_1)$-sparse S_{zero}-matching vector family over $\mathbb{Z}_m$ of length $h+1$.*

Proof. Given a S_{one}-matching vector family $((u_i, v_i))_{i \in [n]}$ of length h, define

$$u_i' = (1, -u_i), \quad v_i' = (1, v_i).$$

As shown in [5] (claim 3.10), this construction satisfies the required inner product conditions for an S_{zero}-matching vector family. We observe that this transformation preserves the sparsity of the original family: for any vector v, the mapping $v \mapsto (1, v)$ adds exactly one non-zero entry to its support modulo p_1. Specifically, $|\operatorname{supp}(v_i' \pmod{p_1})| = |\operatorname{supp}(v_i \pmod{p_1})| + 1$. Similarly, the same argument applied to u_i'. Thus, if the original family is (d, p_1)-sparse, the resulting family is $(d+1, p_1)$-sparse in $\mathbb{Z}_m^{h+1}$. □

By combining the sparse construction of Theorem 3.5 with the S_{zero} transformation Claim 3.6, we obtain a sparse S_{zero}-matching vector family that inherits the parameters of the S_{can} family.

Theorem 3.7. *For every $n, d > 1$, there is a $(d, 2)$-sparse S_{zero}-matching vector family $((u_i, v_i))_{i \in [n]}$ over $\mathbb{Z}_6^h$ with $h \leq d^{O(1)} n^{O(\frac{\log \log d}{\log d})}$.*

Proof. Let $p_1 = 2, p_2 = 3$ distinct primes and $m = p_1 p_2$. Let $((u_i, v_i))_{i\in[n]}$ denote a $(d-1, 2)$-sparse S_{can}-matching vector family over $\mathbb{Z}_m^h$ guaranteed by theorem 3.5. Since $S_{\mathsf{can}} \subseteq S_{\mathsf{one}}$, for $m = 6$, we can apply the transformation from Claim 3.6 and obtain $((u'_i, v'_i))_{i\in[n]}$, a (d, p_1)-sparse S_{zero}-matching vector family over $\mathbb{Z}_m^{h+1}$. □

We say that a collection of matching vectors $((u_i, v_i))_{i\in[n]}$ over $\mathbb{Z}_6^h$ has *computational complexity* of S if there exist a pair of circuits $U, V : \{0,1\}^{\lceil \log n \rceil} \to \mathbb{Z}_6^h$ of size S each, such for every index $i \in [n]$, given in binary representation, the output $U(i)$ (resp., $V(i)$) corresponds to the vector u_i (resp., v_i).[10] We also assume, by default, that the circuits are $\mathrm{poly}(n)$-time uniform. We show that the above constructions of MV have optimal complexity of $O(h)$.

Theorem 3.8. *The MV constructions promised in Theorem 3.5, and consequently those from Theorem 3.7 have computational complexity of $O_d(h)$.*

The "consequently" part follows immediately from the first part by construction, and the proof of the first part is deferred to Section B.

3.3 The LPN Assumption

We rely on the LPN assumption [19, 50] extended to an arbitrary field. We state it here in its dual form, asserting that the syndrome of a uniformly random s-sparse noise vector is pseudorandom.

Definition 3.9 ($\mathbb{F}$ dual-LPN Assumption)**.** *For integer-valued functions $s(\lambda) < t(\lambda) < m(\lambda)$, the (t, m, s)-dLPN assumption over finite field $\mathbb{F}$ asserts that*

$$(H, r) \approx_c (H, He),$$

where $H \xleftarrow{R} \mathbb{F}^{t\times m}$ is a uniformly chosen "parity-check" matrix, $r \xleftarrow{R} \mathbb{F}^t$ is uniformly chosen vector, and $e \xleftarrow{R} \{x \in \mathbb{F}^m \mid wt(x) = s\}$ is a random noise vector of Hamming weight s.[11]

This dual version of LPN is equivalent to the primal version of LPN, i.e., to the pseudorandomness of (C, v) where $C \xleftarrow{R} \mathbb{F}^{m\times m-t}$ is a random generator matrix and $v = c + e$ is a noisy codeword that is sampled by selecting a random vector c in the column span of C and adding a random s-weight noise vector $e \in \mathbb{F}^m$.

Decoding becomes easier as the noise weight decreases and as the redundancy t increases, equivalently, as the primal code rate $(m - t)/m$ decreases. Thus, smaller s and larger t correspond to stronger assumptions. In this work, when we refer to the *LPN assumption*, we mean that dLPN holds over every fixed finite field $\mathbb{F}$, for every constants $C, \epsilon \in (0, 1)$, and for all parameters satisfying

$$m = \lambda, \qquad t > Cm, \qquad s = \lambda^\epsilon.$$

In fact, for our main construction it suffices to assume this only over the fixed ternary field $\mathbb{F}_3$.

Binary-LPN with polynomially-low noise has been studied extensively and used in several cryptographic constructions [4, 44, 61, 43, 1]. Beyond cryptanalytic evidence, binary LPN is supported by several positive forms of evidence, including random-self-reducibility [19] and tight search-to-decision

[10] Here we assume that each entry in $\mathbb{Z}_6$ has some canonical representation in bits.

[11] We use vectors with fixed weight for simplicity. For our constructions, one can use any noise distribution that outputs vectors of weight at most s except with negligible probability (e.g., use iid, Bernoulli-type noise).

reductions [8]. Finite-field variants of LPN have also been studied and used in cryptographic constructions [56, 20, 21, 58, 59, 31]. For fixed-size fields, much of the structural evidence supporting binary LPN extends naturally, with field-size-dependent constant losses. Similarly, known decoding-style attacks on LPN extend from the binary to the q-ary setting with asymptotically comparable complexity when q is constant. Thus, we are not aware of evidence suggesting that LPN over the fixed ternary field $\mathbb{F}_3$ is qualitatively easier than binary LPN.

We note that more aggressive variants appear in the literature, including settings in which the field size grows with the security parameter, the noise weight is polylogarithmic or even slightly super-logarithmic, the rate is polynomially small, or the matrix is sparse.

4 Constructions of RC-PIR

In this section, we prove our main theorem on the existence of short RC-PIR based on the LPN assumption. In Section 4.1, we present a generic transformation that lifts any "nice" PIR scheme to an RC-PIR scheme and analyze it in Section 4.2. In Section 4.3, we construct and analyze a nice PIR scheme based on matching vectors. Finally, in Section 4.4, we combine these ingredients to obtain the final construction.

4.1 RC-PIR from Nice PIR + LPN

A key feature of our model is the capability to transform two-server PIR into an RC-PIR. This capability applies to any two-server PIR whose answer procedure can be expressed as a collection of low-degree multivariate polynomials in the query variables. This algebraic structure will allow us to transform such PIR schemes into RC-PIR schemes.

Notation. Throughout this section, we denote the binomial coefficient $\binom{m+d}{d}$ by $N(m,d)$. This notation would often be used to count the number of monomials in m variables of total degree at most d or the number of m-long vectors over the natural numbers whose ℓ_1-norm is at most d. Note that for a constant d, we have $N(m,d) = O_d(m^d)$.

Theorem 4.1. *Let $\mathbb{F}$ be a finite field. We say that a two-server PIR $\Pi = (\text{QUERY}, \text{ANS}, \text{DEC})$ is d-*nice *over $\mathbb{F}$ if it satisfies the following properties:*

1. *The queries $q_1, q_2 \in \mathbb{F}^t$ and the server's answers $\mathbf{ans}_1, \mathbf{ans}_2 \in \mathbb{F}^\ell$ are vectors over $\mathbb{F}$.*
2. *The answer algorithm $\text{ANS}(x, q)$ can be expressed as ℓ multivariate polynomials of total degree at most d, i.e.,*
$$\text{ANS}(x,q) = \begin{bmatrix} f_x^1(q) \\ \vdots \\ f_x^\ell(q) \end{bmatrix}.$$
3. *The marginal distribution of the first query q_1 is uniform over $\mathbb{F}^t$ and the second query q_2 is generated as a function of the first query q_1 via some function $q_2 = \text{QUERY}_2(i, q_1)$.*

Assuming the $(t = t(n), m = m(n), s = s(n))$-dLPN assumption over $\mathbb{F}$, any nice PIR over $\mathbb{F}$ can be transformed into a two-server RC-PIR scheme with:

1. *Retrieval server communication cost,*

$$\textit{Upload: } N(s,d)\cdot d\log m, \quad \textit{Download: } \ell\cdot N(s,d)\cdot\log|\mathbb{F}|.$$

2. *Computation server communication cost,*

$$\textit{Upload: } t, \quad \textit{Download: } \ell.$$

3. *Retrieval server storage*

$$\ell\cdot N(m,d)\cdot\log|\mathbb{F}|.$$

4. *The RC-PIR query algorithm and decoding algorithm can be implemented by arithmetic circuits over $\mathbb{F}$ of size at most*

$$S(\textsc{Query})+O(mt)+O(N(s,d)\cdot d\log m) \quad \textit{and} \quad S(\textsc{Dec})+O(\ell\cdot N(s,d)),$$

respectively.

Remark 4.2 (The complexity of querying/decoding)**.** *We will keep track of these complexity measures throughout the paper, since they will be important for the MPC application (Theorem 1.4).*

We defer the proof of Theorem 4.1 to Section 4.2. For now we note that the geometric 2PIR of [71] is nice, leading to the following corollary.

Corollary 4.3. *Assuming the $(t=n^{1/3},m,s)$-dLPN assumption, there exists an RC-PIR scheme with communication cost of $O\left(s^3n^{1/3}\right)$ and storage complexity of $O\left(m^3n^{1/3}\right)$. Specifically, by taking $m=2t, s=n^{\epsilon/3}$ we achieve almost cube-root communication of $O(n^{(1/3)+\epsilon})$ and $O\left(n^{4/3}\right)$ storage.*

4.2 Proof of Theorem 4.1

We prove the theorem by presenting a generic construction that applies to any PIR scheme satisfying the above conditions. We first establish a simple useful lemma which we will use in our construction for the encoding and decoding procedures.

Lemma 4.4. *Let $\mathbb{F}$ be a finite field and let $f:\mathbb{F}^m\mapsto\mathbb{F}$ a multivariate polynomial of total degree at most d. If $t\in\mathbb{F}^m$ is an s-sparse vector, then the value of $f(t)$ depends on at most $N(s,d)$ coefficients of f.*

Proof. Let $\mathbb{F}$ be a finite field and $f:\mathbb{F}^m\mapsto\mathbb{F}$ a multivariate polynomial of degree d. Write f in the following form:

$$f(x)=\sum_{\alpha\in\mathbb{N}^m,\sum_{i\in[m]}\alpha_i\le d} c_\alpha x^\alpha \qquad \text{where } x^\alpha=\prod_{j=1}^m x_j^{\alpha_j}.$$

Let $t\in\mathbb{F}^m$ be an s-sparse vector and let its support be

$$E=\{j\in[m]\mid t_j\neq 0\}, \quad |E|=s.$$

Let $\alpha\in\mathbb{N}^m$ such that $\sum_{i\in[m]}\alpha_i\le d$, suppose that there exists some $j\notin E$ with $\alpha_j>0$, then $t_j=0$ and the monomial t^α vanishes:

$$t^\alpha=\prod_{j=1}^m t_j^{\alpha_j}=0.$$

Hence, only monomials with $\mathrm{supp}(\alpha) \subseteq E$ can contribute to the value of $f(t)$ and the value may be calculated in the following way:

$$f(t) = \sum_{\substack{\alpha \in \mathbb{N}^m \\ \mathrm{supp}(\alpha) \subseteq E \\ \sum_{i \in [m]} \alpha_i \leq d}} c_\alpha t^\alpha$$

where the number of coefficients c_α in the sum is exactly $N(s, d)$. □

Construction 4.5 (RC-PIR from Nice PIR)**.** *Consider any two-server PIR scheme* $(\textsc{Query}, \textsc{Ans}, \textsc{Dec})$ *satisfying the assumptions of Theorem 4.1. Under these assumptions, there exists a deterministic function* $\textsc{Query}_2$ *such that for any index* i*:*

$$\textsc{Query}(i; r) \equiv (q_1, \textsc{Query}_2(i, q_1)),$$

where $q_1 = r$ *is sampled uniformly from* $\mathbb{F}^t$*. We construct an RC-PIR scheme as follows. Let* $m = m(t)$ *be the number of samples in public random matrix* $H \in \mathbb{F}^{t \times m}$ *and* $s = s(t)$ *the sparsity setting. Sample a random public matrix* $H \in \mathbb{F}^{t \times m}$*. Then under the* (t, m, s)*-dLPN assumption we define the following RC-PIR scheme.*

1. *The* $\textsc{Encoder}(x)$ *algorithm encodes the database the following way. For any* $j \in [\ell]$ *and polynomial* $f_x^j(q)$ *we define a new polynomial* $g_x^j(e) = f_x^j(He)$ *where* $e \in \mathbb{F}^m$*. The polynomial* $g_x^j(e)$ *is a degree-*d *multivariate polynomial over* m *variables, thus have at most* $N(m, d)$ *monomials. Rewrite this polynomial as follows:*

$$g_x^j(e) = \sum_{\alpha \in \mathbb{N}^m, \sum_{i \in [m]} \alpha_i \leq d} c_\alpha e^\alpha \qquad \text{where } e^\alpha = \prod_{j=1}^{m} e_j^{\alpha_j}.$$

   Store all coefficients c_α *into* R^j *such that* $R^j_\alpha = c_\alpha$ *where* α *is the entry's address and outputs*

$$R = (R^j)_{j \in [\ell]}.$$

   Note that the address space is the set of all m*-long vectors over the natural numbers whose* ℓ_1*-norm is at most* d*. Thus, there are exactly* $B = N(m, d)$ *possible addresses. For every address* α*, the stored value is the* ℓ*-tuple*

$$(R^1_\alpha, \ldots, R^\ell_\alpha) \in \mathbb{F}^\ell,$$

   containing the coefficients of the ℓ *answer polynomials associated with the monomial indexed by* α*. Hence, each block contains* ℓ *field elements and the encoded database consists of* $N(m, d)$ *blocks.*[12]

2. *The query algorithm* $\textsc{Query}(i; \rho)$ *given the randomness samples an* s*-sparse vector* $e \in \mathbb{F}^m$ *uniformly and outputs a set of requested entries*

$$q_r = \left\{ \alpha \in \mathbb{N}^m \mid \mathrm{supp}(\alpha) \subseteq \mathrm{supp}(e), \sum_{j \in [m]} \alpha_i \leq d \right\}, \qquad q_c = \textsc{Query}_2(i, He).$$

   In other words, q_r *represents a set of coefficients* c_α *the client wants to read from* S_R*. Those coefficients will be used to calculate the output of the answer polynomials on* e*.*

[12] We treat the public matrix H as a public global parameter that can be reused in various systems and so we do not count it as client storage. Note that after the encoding step the server does not need to keep it in memory.

3. *The retrieval server given a query to read coefficient α (entry address) executes the algorithm* $\text{RANSWER}(R, \alpha)$ *which outputs*

$$ans_r^\alpha = \begin{bmatrix} R_\alpha^1 \\ \vdots \\ R_\alpha^\ell \end{bmatrix} \in \mathbb{F}^\ell.$$

4. *The computation server is left unchanged, given its query q_c executes the algorithm* $\text{CANSWER}(x, q_c)$ *that outputs* $\mathbf{ans}_c = \text{ANS}(x, q_c)$.

5. *The decoding algorithm* $\text{DEC}^{rc}(i, \mathbf{ans}_r, \mathbf{ans}_c)$ *given the index $i \in [n]$ and the server responses* $(\{\mathbf{ans}_r^\alpha\}, \mathbf{ans}_c)$*, the decoder uses* $\mathbf{ans}_r = \{\mathbf{ans}_r^\alpha\}$ *to reconstruct the values*

$$\mathbf{ans}'_r = \begin{bmatrix} g_x^1(e) \\ \vdots \\ g_x^\ell(e) \end{bmatrix} = \begin{bmatrix} f_x^1(He) \\ \vdots \\ f_x^\ell(He) \end{bmatrix} = \text{ANS}(x, He)$$

since, by Lemma 4.4, $\mathbf{ans}_r$ *contains exactly the coefficients required to evaluate $g_x^j(e)$. Finally, the decoder runs the original PIR decoding procedure on the reconstructed answer*

$$\text{DEC}(i, \mathbf{ans}'_r, \mathbf{ans}_c).$$

The correctness follows directly from Lemma 4.4 and the correctness of the original PIR protocol.

4.2.1 Privacy of Construction 4.5

By definition of the original PIR privacy for every pair of indices $i, j \in [n]$ when query $q_1 \xleftarrow{R} \mathbb{F}^t$ is sampled uniformly the following two distributions

$$q_2 = \text{QUERY}_2(i, q_1) \qquad \text{and} \qquad q'_2 = \text{QUERY}_2(j, q_1)$$

are identical. In the RC-PIR construction, the retrieval server receives a query that is independent of the index i. Hence, the retrieval server's view is identically distributed for all indices and privacy in respect to S_R follows immediately.

For the computation server, its view is

$$(x, H, \text{QUERY}_2(i, He)),$$

where $e \in \mathbb{F}^m$ is sampled uniformly among all s-sparse vectors and x is independent of $H, \text{QUERY}_2(i, He)$. Hence, under the (t, m, s)-dLPN assumption for any index $i \in [n]$,

$$(x, H, \text{QUERY}_2(i, He)) \approx_c (x, H, \text{QUERY}_2(i, q_1))$$

where $q_1 \xleftarrow{R} \mathbb{F}^t$ is distributed uniformly. By the privacy of the original PIR scheme, for every pair of indices $i, j \in [n]$,

$$\text{QUERY}_2(i, q_1) \equiv \text{QUERY}_2(j, q_1).$$

Combining the two equations, we obtain that for any pair of indices $i, j \in [n]$,

$$(x, H, \text{QUERY}_2(i, He)) \approx_c (x, H, \text{QUERY}_2(j, He)),$$

where $e \in \mathbb{F}^m$ is sampled uniformly among all s-sparse vectors.

4.2.2 Analyzing the complexity of the construction

We now analyze the complexity of Construction 4.5. The retrieval server storage maps each monomial coefficient into an entry in the processed database, while each entry consists of ℓ coefficients one for each polynomial of the answer algorithm, this requires

$$\ell \cdot N(m,d) \cdot \log|\mathbb{F}| \in \mathrm{poly}(n)$$

bits of storage. We next consider the communication cost. The retrieval server's query explicitly defines the $N(s,d)$ entries the client downloads, while each entry address may be represented using at most $d\log m$ bits. Hence, overall the retrieval server receives

$$N(s,d) \cdot d\log m$$

bits and answers with

$$\ell \cdot N(s,d) \cdot \log|\mathbb{F}|$$

bits. This asymptotically improves the original PIR query communication cost of the retrieval server but with some overhead to the answer communication cost. The computation server communication remains unchanged.

For computational complexity, the new query-generation algorithm samples an s-sparse vector $e \in \mathbb{F}^m$, computes He, invokes the original query algorithm, and generates the retrieval query q_r. The multiplication by $H \in \mathbb{F}^{t\times m}$ contributes $O(mt)$ to the circuit size.[13] In addition, the algorithm enumerates all $N(s,d)$ monomial addresses supported on $\mathrm{supp}(e)$. Using the compressed representation in which each address is specified by at most d indices in $[m]$, each address has length $O(d\log m)$. Since Boolean operations can be emulated by constant-size arithmetic circuits over $\mathbb{F}$, computing and writing the address list contributes at most $O(N(s,d)\cdot d\log m)$ additional gates.

The RC-PIR decoding algorithm first reconstructs the base PIR answer $\mathbf{ans}'_r$ from the retrieved coefficients $\mathbf{ans}_r$, and then applies the original decoding algorithm. By Lemma 4.4, the reconstruction of each coordinate of $\mathbf{ans}'_r$ requires $O(N(s,d))$ additions. Since $\mathbf{ans}'_r$ consists of ℓ coordinates, the total reconstruction cost is $O(\ell N(s,d))$, leading to the claimed decoding circuit size.

4.3 Nice PIR based on Matching Vectors

We construct nice-PIR based on sparse Matching Vectors. We will prove the following theorem.

Theorem 4.6 (Nice-PIR from sparse-MVs)**.** *For every constant d, there exists a two-server PIR (with negligible decoding error) which is $2d$-nice over $\mathbb{Z}_3$. The upload/download complexity and the computational complexity of the decoding and query algorithms are all $\mathrm{polylog}(n)d^{O(1)}n^{\epsilon}$, where $\epsilon = O(\frac{\log\log d}{\log d})$.*

4.3.1 The Sparse MV RC-PIR scheme

We begin by describing a nice PIR protocol over $\mathbb{Z}_3$ whose decoding algorithm fails (i.e., outputs a special abort symbol) with non-negligible probability, and then show how to reduce this error via repetition. We start by introducing the necessary notation.

[13] In the RAM model we can reduce this cost to $O(s(m+t))$ by exploiting the sparsity.

Casting operation. We will need the following technical mod-3 to mod-2 casting operations. First let $\phi_2 : \mathbb{Z}_3 \mapsto \{0,1\}$ be a mapping that takes 0/1 to themselves, and 2 to 0. In particular, this can be written as

$$\phi_2(x) = x(2-x) \pmod 3.$$

This mapping is naturally extended to vectors over $\mathbb{Z}_3$. Next, given a pair of vectors $a, b \in \mathbb{Z}_3^D$ we would like to define a polynomial $\mathrm{MP}_2(a,b)$ whose output is exactly the binary inner-product applied to the vectors $a' = \phi_2(a)$ and $b' = \phi_2(b)$. Being a function over $\mathbb{Z}_3^{2D}$ variables, $\mathrm{MP}_2(a,b)$ can be written as a polynomial of degree at most $4D$. (We will later see that, in our case, this can be slightly optimized.)

Construction 4.7 (Nice Sparse Matching Vector based PIR)**.** *Let $d > 1$ denote the sparsity constant, $x \in \{0,1\}^n$ denote a database of size n and $((u_i, v_i))_{i\in[n]}$ be a $(d,2)$-sparse S_{zero}-matching vector family such that $u_i, v_i \in \mathbb{Z}_6^h$ for all $i \in [n]$ (as promised by Theorem 3.7). We let $\hat{u}_i = u_i \bmod 2$ and $\hat{v}_i = v_i \bmod 2$.*

1. *Query Generation: Given the index i, the client samples a uniformly random vector $r \in \mathbb{Z}_3^h$ and sends to server $s \in \{0,1\}$ the query $q_s = r + s \cdot \hat{u}_i \pmod 3$. We call the query* good *if for all $\ell \in [h]$, the* integer *sum satisfies*
$$r[\ell] + (\hat{u}_i[\ell]) \le 2.$$
2. *Answer: Server s, given a query $q \in \mathbb{Z}_3^h$, output*
$$\mathbf{ans}_s = \sum_{j\in[n]} \mathrm{MP}_2(q, \hat{v}_j) \cdot x_j \cdot v_j \pmod 3.$$
3. *Decoding: If the query is not good, output a failure symbol $\perp$. Otherwise, given the index i and the answers of the servers, outputs 1 if*
$$\langle u_i, \mathbf{ans}_1 - \mathbf{ans}_0 \rangle \not\equiv 0 \pmod 3$$
and 0 otherwise.

Claim 4.8 (Correctness of Construction 4.7)**.** *For every database $x \in \{0,1\}^n$, sparsity parameter $d > 0$, and every index $i \in [n]$, consider the randomness r of the client and let $(q_1, q_2) = \textsc{Query}(i; r)$, then conditioned on not aborting, the server does not err, i.e.,*

$$\textsc{Dec}(i, r, \textsc{Ans}(x, q_1), \textsc{Ans}(x, q_2)) = x_i.$$

Moreover, the probability of not aborting is at least $\left(\frac{2}{3}\right)^d$.

Proof. By definition, the decoder does not abort if the queries are good, i.e., if for all $\ell \in [h]$, the integer sum satisfies

$$r[\ell] + \hat{u}_i[\ell] \le 2,$$

which happens with probability at least $\left(\frac{2}{3}\right)^d$ (due to the d-sparsity of u).

We show that if the queries are good the decoder does not err. The crucial observation is that if the queries are good then $q_1 = r + \hat{u}_i \pmod 3$ satisfies the same equation modulo-2, i.e.,

$$q_1 - r = \hat{u}_i \pmod 2. \tag{2}$$

This allows us to expand the decoding expression as follows. (For brevity, we let $\langle \cdot, \cdot \rangle_2$ denote inner-product mod-2.)

$$\begin{aligned}
\langle u_i, \mathbf{ans}_1 - \mathbf{ans}_0 \rangle &= \sum_{j \in [n]} \mathrm{MP}_2(q_1, \hat{v}_j) \cdot x_j \cdot \langle u_i, v_j \rangle \\
&\quad - \sum_{j \in [n]} \mathrm{MP}_2(r, \hat{v}_j) \cdot x_j \cdot \langle u_i, v_j \rangle \pmod 3 \\
&=_* \sum_{j \in [n]} \langle q_1, \hat{v}_j \rangle_2 \cdot x_j \cdot \langle u_i, v_j \rangle \\
&\quad - \sum_{j \in [n]} \langle r, \hat{v}_j \rangle_2 \cdot x_j \cdot \langle u_i, v_j \rangle \pmod 3 \\
&= \sum_{j \in [n]} (\langle q_1, \hat{v}_j \rangle_2 - \langle r, \hat{v}_j \rangle_2) \cdot x_j \cdot \langle u_i, v_j \rangle \pmod 3
\end{aligned}$$

where the transition * relies on (2). We analyze each term separately based on the properties of the $(d, 2)$-sparse S_{zero}-matching vector family. For $j \neq i$, by the matching vector property, either $\langle u_i, v_j \rangle = 0 \pmod 2$ or $\langle u_i, v_j \rangle = 0 \pmod 3$. We first claim that for $i \neq j$,

$$(\langle q_1, \hat{v}_j \rangle_2 - \langle r, \hat{v}_j \rangle_2) \cdot x_j \cdot \langle u_i, v_j \rangle \pmod 3 = 0.$$

Clearly, this is true if $\langle u_i, v_j \rangle = 0 \pmod 3$. Otherwise,

$$\langle u_i, v_j \rangle = 0 \pmod 2 \implies \langle \hat{u}_i, \hat{v}_j \rangle_2 = 0,$$

thus

$$\langle q_1, \hat{v}_j \rangle_2 - \langle r, \hat{v}_j \rangle_2 = \langle r + \hat{u}_i, \hat{v}_j \rangle_2 - \langle r, \hat{v}_j \rangle_2 = \langle r, \hat{v}_j \rangle_2 - \langle r, \hat{v}_j \rangle_2 = 0 \pmod 3$$

In both cases, the term vanishes. It is left to show that the coefficient of x_i is non-zero. That is, we show that the term

$$(\langle q_1, \hat{v}_i \rangle_2 - \langle r, \hat{v}_i \rangle_2) \cdot \langle u_i, v_i \rangle \pmod 3$$

is non-zero. The second multiplicand is non-zero by definition ($\langle u_i, v_i \rangle \not\equiv 0 \pmod 3$). To show that the first multiplicand is non-zero it suffices to show that $\langle q_1, \hat{v}_i \rangle_2 \neq \langle r, \hat{v}_i \rangle_2$. Indeed, recalling that $\langle u_i, v_i \rangle \not\equiv 0 \pmod 2$, we have

$$\begin{aligned}
\langle q_1, \hat{v}_i \rangle_2 &= \langle r + \hat{u}_i, \hat{v}_i \rangle_2 \\
&= \langle u_i, \hat{v}_i \rangle + \langle r, \hat{v}_i \rangle \pmod 2 \\
&= 1 + \langle r, \hat{v}_i \rangle \pmod 2 \\
&\neq \langle r, \hat{v}_i \rangle_2,
\end{aligned}$$

Thus, the decoder correctly outputs x_i. □

Claim 4.9 (Security of Construction 4.7)**.** *For every pair of indices $i, j \in [n]$ and every server $s \in \{0, 1\}$ the queries q_0 (resp, q_1) is uniformly distributed over $\mathbb{Z}_3^h$.*

Proof. Since $q_0 = r$ is uniformly sampled from $\mathbb{Z}_3^h$. Moreover, because r is a uniform random variable and the vector $\hat{u}_i = (u_i \bmod 2)$ is a constant independent of r, their sum q_1 is also uniformly distributed over $\mathbb{Z}_3^h$. □

Niceness. Clearly, the queries are vectors over $\mathbb{Z}_3$, the marginal distribution of the first query is uniform, and the second query is obtained deterministically from the first query by an efficient function. Thus, it remains to bound the degree of the answer polynomials. For this purpose, we use the fact that in every term $\mathrm{MP}_2(q, \hat{v}_j)$, the vector $\hat{v}_j \in \{0,1\}^h$ is fixed and has support of size at most d. For a fixed $b \in \{0,1\}^h$ with $I = \mathrm{supp}(b)$, the function $q \mapsto \mathrm{MP}_2(q, b)$ can be written over $\mathbb{Z}_3$ as

$$\mathrm{MP}_{2,b}(q) = 2\left(1 - \prod_{\ell \in I}(1 - 2\phi_2(q_\ell))\right) \pmod 3.$$

Since ϕ_2 has degree 2 and $|I| \leq d$, this polynomial has degree at most $2d$. It follows that, for every fixed database x, each coordinate of the answer map

$$q \mapsto \sum_{j \in [n]} \mathrm{MP}_2(q, \hat{v}_j) \cdot x_j \cdot v_j \pmod 3$$

is a polynomial over $\mathbb{Z}_3$ of degree at most $2d$. Therefore, the construction is $2d$-nice.

Computational complexity. The computational complexity of the query and answering algorithm is $O(h)$ plus the complexity of the MV family (i.e., given i output v_i and u_i) which is $O_d(h)$ (as shown in Theorem 3.8). So the total complexity is $O_d(h)$.

Amplifying the Success Probability. The success probability of the decoding algorithm can be amplified through parallel repetition. If the protocol is executed T independent times with the same requested index i, the error probability (that *all* queries are bad) is exactly

$$\left(1 - \left(\frac{2}{3}\right)^d\right)^T.$$

Since d is constant we can push reduce the error probability to $\mathrm{negl}(n)$ by taking $T = \mathrm{polylog}(n)$. This transformation increases the complexity (communication/computation) by a factor of T and does not affect the niceness property. Theorem 4.6 follows.

4.4 Putting it all together

By combining Theorem 4.6 with Theorem 4.1, we derive the following theorem. (Recall that $N(m,d) = \binom{m+d}{d}$.)

Theorem 4.10. *For every positive constant d, there exist $\epsilon_d = O(\frac{\log\log d}{\log d})$ and $h = d^{O(1)} n^{\epsilon_d}$ such that, under the (h, m, s)-dLPN assumption over $\mathbb{Z}_3$, there exists an RC-PIR scheme with negligible decoding error in which:*

1. *The computation server has upload/download complexity* $\mathrm{polylog}(n)h$.
2. *The retrieval server has upload complexity*

$$\mathrm{polylog}(n) \cdot N(s,d) \cdot d \log m = O_d(\mathrm{polylog}(n) s^d \log m)$$

and download complexity

$$\mathrm{polylog}(n) \cdot h \cdot N(s,d) = O_d(\mathrm{polylog}(n) h s^d).$$

3. *The retrieval server storage is $O\big(h \cdot N(m,d)\big) = O_d(hm^d)$ field elements.*

4. *The query and decoding algorithms can be implemented by circuits of size*

$$\mathrm{polylog}(n)h + O(mh) + O(N(s,d) \cdot d \log m) = \mathrm{polylog}(n)h + O(mh) + O_d(s^d \log m)$$

and

$$\mathrm{polylog}(n)h + O(hN(s,d)) = \mathrm{polylog}(n)h + O_d(hs^d),$$

respectively.

In this theorem, d denotes the degree/niceness parameter of the nice PIR scheme obtained from Theorem 4.6. This differs from the sparsity parameter in the underlying matching-vector construction by only a constant factor, and therefore does not affect the asymptotic bound $\epsilon_d = O\Big(\frac{\log\log d}{\log d}\Big)$.

Corollary 4.11 (Thm. 1.1 restated)**.** *Assuming dLPN, for every constant $\epsilon > 0$, there exists a two-server RC-PIR whose communication complexity and the circuit size of querying/decoding is at most $O(n^{\epsilon})$ and where the storage complexity is $O(n^{c_\epsilon})$ where n is the length of the database and the constant $c_\epsilon = \epsilon \exp(O(\epsilon^{-1}\log(\epsilon^{-1})))$.*

Proof. Given $\epsilon > 0$, apply Thm. 4.10 as follows. Choose a constant d so that $\epsilon_d < \epsilon/2$ and set the dLPN parameters to $m = 2h$ and $s = n^{\delta/d}$ for constant $\delta < \epsilon/2$. This yields total communication complexity $\tilde{O}_d(n^{\epsilon_d+\delta}) = O(n^{\epsilon})$, storage complexity $O_d(n^{\epsilon_d(d+1)})$, and querying/decoding circuit sizes of at most $O_d(n^{2\epsilon_d}) + \tilde{O}_d(n^{\delta}) = O(n^{\epsilon})$ and $O_d(n^{\epsilon_d+\delta}) = O_d(n^{\epsilon})$, respectively.[14]. Note that we can write $d = \exp(O(\epsilon^{-1}\log(\epsilon^{-1})))$ and so the storage can be written as n^{c_ϵ} where $c_\epsilon = \epsilon\exp(O(\epsilon^{-1}\log(\epsilon^{-1})))$. □

5 Applications

5.1 From 2-server RC-PIR to fully-secure k-server PIR

In this section, we show how to convert a two-server RC-PIR into a k-server PIR with $k-1$-out-of-k privacy (aka full-privacy).

We state our basic extension-theorem in a refined way. That is, we assume that given a database with n elements, the RC-PIR stores an encoded database $R \in \{0,1\}^{\ell(n)\times B(n)}$ where ℓ is the block size and B is the number of blocks. Such an RC-PIR will be referred to as having a storage of $\ell(n) \times B(n)$ elements. We refer to the *retrieval complexity*, $t_r(n)$ of an RC-PIR as the number of blocks that the client reads from the retrieval servers per query. That is, the upload communication cost of the retrieval server is $t_r \log |B|$ bits and the download communication complexity is $t_r \ell \log |B|$ elements.

Theorem 5.1. *Let Π_k be a fully private k-server RC-PIR scheme and let Π_2 be a 2-server RC-PIR. Assume that Π_k (resp., Π_2) has storage complexity of $\ell_k(n) \times B_k(n)$ (resp., $\ell_2(n) \times B_2(n)$), retrieval complexity of $\rho_k(n)$ (resp., $\rho_2(n)$), and that the total upload and download communication cost of the computation servers are $U_k(n)$ and $D_k(n)$ (resp., U_2 and D_2). Let $S_k(n)$ (resp., $S_2(n)$) denote an upper-bound on the circuit size of $\mathrm{QUERY}_k(1^n,\cdot)$ and $\mathrm{DEC}_k(1^n,\cdot)$ (resp., $\mathrm{QUERY}_2(1^n,\cdot)$ and $\mathrm{DEC}_2(1^n,\cdot)$).*

[14]The $\tilde{O}$ notation suppresses logarithmic factors. Specifically, $g(n) = \tilde{O}(f(n))$ if there exists a positive constant $C > 0$ such that $g(n) \le Cf(n) \cdot \log^C(n)$ for all sufficiently large n's. The O_d notation indicates that the hidden constant in the Big-O bound may depend on d. That is, $g(n) = O_d(f(n))$ if $g(n) = O(h(d) \cdot f(n))$ for some function $h(\cdot)$.

Then there exists a fully private $(k+1)$-server RC-PIR scheme Π_{k+1} with storage complexity of $\ell_k(n) \cdot \ell_2(B_k(n)) \times B_2(B_k(n))$, retrieval complexity of $\rho_k(n) \cdot \rho_2(B_k(n))$, and where the computation servers have total upload and download communication cost of $U_k(n)+\rho_k(n)U_2(B_k(n))$ and $D_k(n)+\rho_k(n)\ell_k(n)D_2(B_k(n))$. Moreover, the query algorithm and decoding algorithms of Π_{k+1} can be implemented by circuits of size at most $S_k(n) + \rho_k(n) \cdot S_2(B_k(n))$.

The core idea behind Theorem 5.1 is to emulate the retrieval server of Π_k using the 2-server PIR scheme Π_2. Specifically, we first encode x into an $\ell_k \times B_k$ database R' using the encoder $\text{ENCODER}_k(x)$ of Π_k. We then treat each row R'_i of R' as a separate database and re-encode it using the encoder $\text{ENCODER}_2(R'_i)$ of Π_2. The final encoded database is the matrix R obtained by stacking all resulting matrices one on top of another. The queries to the first $k-1$ servers, as well as their responses, are generated exactly as in Π_k. The queries to the final two servers are generated by first computing the kth "fetch" queries of the retrieval server and then interpreting each such query as a PIR request (over Π_2) to each of the databases $R'_1, \ldots, R'_{\ell_k(n)}$. Full details are deferred to Section C.

Overall, we conclude the following high-level proposition for the case of RC-PIR with $1 \times n^\beta$ storage, aka n^β-storage RC-PIR.[15]

Proposition 5.2. *Let $\epsilon, \epsilon' \in (0,1)$ and $c, c' \geq 1$ be constants. Given a fully-private k-server RC-PIR Π_k with communication cost n^ϵ and storage complexity of n^c and 2-server RC-PIR Π_2 with communication cost $n^{\epsilon'}$ and storage complexity $n^{c'}$, the $(k+1)$-server fully-private RC-PIR Π_{k+1} has storage complexity of $n^{cc'}$, and communication cost of of at most $O(n^{\epsilon+c\epsilon'})$. Furthermore, assuming that the circuit complexity of querying/decoding in Π_k and Π_2 are at most n^ϵ and $n^{\epsilon'}$, respectively, the circuit complexity of querying/decoding in Π_{k+1} is at most $O(n^{\epsilon+c\epsilon'})$.*

Proof. By Theorem 5.1, the storage complexity of Π_{k+1} is $n^{c \cdot c'}$. The upper-bound on the communication allows us to upper-bound $U_k(n)$, $D_k(n)$ and $\rho_k(n)\log(n^c)$ by n^ϵ. Similarly, we upper-bound $U_2(n)$, $D_2(n)$ and $\rho_2(n)\log n^{c'}$ by $n^{\epsilon'}$. Therefore, in Π_{k+1} the retrieval server retrieves at most $\rho_k(n) \cdot \rho_2(B_k(n))$ entries and so it communicates (upload and download) at most

$$O(\log(n^{cc'}) \cdot \rho_k(n) \cdot \rho_2(B_k(n))) \leq O(n^{\epsilon+c\epsilon'})$$

bits. In addition, and total (upload + download) communication cost of the computation servers is at most

$$U_k(n) + \rho_k(n)U_2(B_k(n)) + D_k(n) + \rho_k(n)D_2(B_k(n)) \leq n^\epsilon + n^{\epsilon+c\cdot\epsilon'} + n^\epsilon + n^{\epsilon+c\epsilon'} = O(n^{\epsilon+c\epsilon'}),$$

as claimed. The computational complexity follows by a similar calculation. □

We derive the following corollary (whose last part is a restatement of Theorem 1.2).

Corollary 5.3 (from short RC-PIR to short fully-private k-PIR)**.** *Assume the existence of an RC-PIR Π_2 with short communication, that is, for every $\epsilon > 0$, there exists an RC-PIR with communication n^ϵ, querying/decoding complexity of n^ϵ, and storage of n^c for some $c = c(\epsilon)$. Then, for every constant $k \geq 2$ and $\delta > 0$, there exists a k-server RC-PIR Π_k with full privacy, communication of n^δ, querying/decoding complexity of n^δ, and polynomial storage of n^b for some constant $b = b(k, \delta)$. Specifically, the implication holds under the LPN assumption.*

[15] Note that we can always view an RC-PIR with storage of $\ell(n) \times B(n)$ as a $1 \times \ell(n)B(n)$-storage RC-PIR at the expense of increasing the upload cost of the retrieval server by a factor of $\ell(n)$.

Proof. We prove the existence of such a scheme by induction on the number of servers $k \geq 2$. The base case, $k = 2$, follows by the assumption on Π_2. To prove the induction step, we show how to construct a fully-private $(k+1)$-server RC-PIR with target communication cost n^δ. By the induction hypothesis, there exists (1) a fully-private k-server RC-PIR Π_k with communication/querying/decoding cost n^ϵ and storage complexity n^c, where $\epsilon = \frac{\delta}{2}$ and $c = c(\epsilon)$; and (2) a fully-private 2-server RC-PIR Π_2 with communication/querying/decoding cost $n^{\epsilon'}$ and storage complexity $n^{c'}$, where $\epsilon' = \frac{\delta}{2c}$ and $c' = c'(\epsilon')$. By Proposition 5.2, the resulting $(k+1)$-server scheme is fully private and has communication/querying/decoding cost of $n^{\epsilon + c \cdot \epsilon'} \leq n^\delta$, and storage complexity of $n^{cc'}$. Finally, the "specifically" part follows by combining the first part with Corollary 4.11. □

5.2 PIR for General Access Structures

PIR over general access structure. A k-access structure is an upward-closed collection (aka monotone) collection $\Gamma \subseteq 2^{[k]}$ of authorized subsets, i.e., if $A \in \Gamma$ and $A \subseteq B \subseteq [k]$, then $B \in \Gamma$. A k-server PIR $(\text{QUERY}, \text{ANS}, \text{DEC})$ realizes Γ if correctness holds for sets in Γ and privacy holds for sets not in Γ. Formally, the correctness requirement asserts that for every x, i and $A \in \Gamma$, it holds that

$$\Pr_r[\text{DEC}(i, r, A, (\mathbf{ans}_j : j \in A)) \neq x_i] \leq \text{negl}(n),$$

where

$$(q_1, q_2, \ldots, q_k) = \text{QUERY}(i; r) \quad \text{and} \quad \mathbf{ans}_j = \text{ANS}(j, x, q_j), \forall j \in [k].$$

The privacy requirement that for every i, j and $T \notin \Gamma$

$$(\text{QUERY}(1^n, i; r))_T \approx_c (\text{QUERY}(1^n, j; r))_T, .$$

The definition also extends to a partial access structure; roughly, in this case Γ is replaced with two disjoint collections, a downward-closed set Γ_0 and an upward-closed set Γ_1, and correctness (resp., privacy) should hold for every set in Γ_1 (resp., Γ_0). Specifically, k-server t-private PIR corresponds to the partial k-access structure given by $\Gamma_0 = \{A : |A| \leq t\}$ and $\Gamma_1 = \{[k]\}$. A k-server $(k-1)$-private PIR (aka fully-private PIR) corresponds to the (fully-defined) access structure Γ that contains the single set $[k]$.

Lemma 5.4 (RC-PIR for general access structures)**.** *Every finite access structure Γ over $k \geq 2$ parties can be realized by a PIR Π_Γ with communication complexity of at most $2^k C_k(n)$ where n is the database size and $C_k(n)$ denotes the communication complexity of k-server fully-private PIR Π_k over n-size database. Similarly, the complexity of computing a query (resp., computing an answer, decoding the output) in Π_Γ is at most $2^k f_k(n)$ where $f_k(n)$ is the complexity of computing a query (resp., computing an answer, decoding the output) in Π_k over n-size database.*

Proof. Let $S_1, \ldots, S_L \subseteq [k]$ be the minimal authorized sets of Γ. For each $\ell \in [L]$, let $k_\ell = |S_\ell|$, and let $S_{\ell,j}$ denote the jth element of S_ℓ under some fixed canonical ordering. For every $\ell \in [L]$, let $\Pi_\ell = (\text{QUERY}_\ell, \text{ANS}_\ell, \text{DEC}_\ell)$ be a k_ℓ-server fully private PIR scheme. Since $k_\ell \leq k$, we may assume that the communication complexity of Π_ℓ is at most $C_k(n)$.[16]

[16] Given a fully private k-server PIR scheme, one can obtain a fully private k'-server PIR scheme for every $k' \leq k$ with the same total communication complexity, for example by sending the queries intended for the last $k - k'$ servers to the first server, or by distributing them arbitrarily among the k' servers.

We now define a k-server PIR scheme Π realizing Γ. Given an index $i \in [n]$, the client independently samples randomness r_ℓ for every $\ell \in [L]$, generates query tuples

$$q_\ell = (q_{\ell,1}, \ldots, q_{\ell,k_\ell}) \xleftarrow{R} \text{QUERY}_\ell(i; r_\ell),$$

and sends query $q_{\ell,j}$ to server $S_{\ell,j}$.

For every query $q_{\ell,j}$ received by server $S_{\ell,j}$, the server responds according to Π_ℓ with

$$\mathbf{ans}_{\ell,j} = \text{ANS}_\ell(S_{\ell,j}, x, q_{\ell,j}).$$

Suppose the decoder receives the answers of all servers in some authorized set $A \subseteq [k]$. It first finds a minimal authorized set $S_\ell \subseteq A$, retrieves the corresponding answer vector

$$\mathbf{ans}_\ell = (\mathbf{ans}_{\ell,j})_{j \in [k_\ell]},$$

and recovers x_i by applying the decoder $\text{DEC}_\ell(i, r_\ell, \mathbf{ans}_\ell)$.

Correctness is immediate. To prove privacy, fix an unauthorized set $T \subseteq [k]$. For every $\ell \in [L]$, define

$$T_\ell = T \cap S_\ell.$$

Since T is unauthorized for Γ, the set T_ℓ is unauthorized in the PIR scheme Π_ℓ. For an index $i \in [n]$ and an index $i' \in [n]$, the queries observed by T in the ℓth scheme are

$$\text{QUERY}_\ell(i)[T_\ell] \quad \text{and} \quad \text{QUERY}_\ell(i')[T_\ell],$$

respectively. By the privacy of Π_ℓ, these distributions are indistinguishable for every $\ell \in [L]$. Moreover, these random variables are mutually independent, since the randomness values $r_1, \ldots, r_L$ are sampled independently. It follows that the overall views of T corresponding to indices i and i' are indistinguishable, establishing privacy. □

Remark 5.5 (A formula-based generalization)**.** *A natural approach for realizing a general access structure Γ is to represent Γ by a monotone formula and realize it recursively. The idea is to view each wire i as a "virtual" server S_i. Then, for every gate g with outgoing wire i and incoming wires j, ℓ, we replace the virtual server S_i with a PIR scheme over the servers S_j, S_ℓ that realizes the access structure induced by the gate g.*

While this paradigm works well for secret sharing and general MPC (e.g., [57, 52, 36])), it becomes problematic in the context of PIR. Indeed, although we can realize both the AND access structure (via a standard 2-server PIR) and the trivial OR access structure (by replicating the database and sending the query in the clear), recursion fails after an AND gate. The reason is that in a 2-server PIR corresponding to an AND gate, each server performs a "compute" operation, which cannot itself be implemented as a PIR "retrieve" operation in the next recursive layer.

This obstacle disappears once we have a 2-server RC-PIR scheme. Such a scheme realizes the AND access structure while still allowing recursion on one of the incoming wires - namely, the wire corresponding to the retrieval server. Consequently, we can realize every monotone formula over AND/OR gates in which each AND gate has at least one child that is AND-free (that is, an OR-tree over input wires).

For example, one can use a chain of such AND gates to realize a k-ary AND gate (as implicitly done in Thm. 5.1), and then use these k-ary AND gates to realize an arbitrary DNF formula (as in Lemma 5.4). Other decompositions are also possible, such as CNF representations, and may yield better concrete efficiency for certain access structures.

By combining Lemma 5.4 with Corollary 5.3, we derive the following corollary whose last part is a restatement of Theorem 1.3.

Corollary 5.6 (from short RC-PIR to short general-PIR)**.** *Assume that for every $\epsilon > 0$, there exists an RC-PIR with communication/query/decoding complexity of n^ϵ, and polynomial storage of n^c for some $c = c(\epsilon)$.*

Then, for every constant $k \geq 2$, every monotone access structure Γ over k servers, and every constant $\epsilon > 0$, there exists a k-server PIR protocol realizing Γ with communication/query/decoding complexity of $O(2^k n^\epsilon)$. Specifically, the implication holds under the dLPN assumption.

5.3 PIR with Byzantine Servers

Once we have PIR for general access structure, we can use it to handle the case of Byzantine servers. The proof follows the argument of [52] (in the context of general MPC) and is based on their notion of Q_3 *access structure*. Formally, an access structure Γ over k servers is Q_3 if there do not exist unauthorized sets $A_1, A_2, A_3 \subseteq [k]$, with $A_i \notin \Gamma$, such that $A_1 \cup A_2 \cup A_3 = [k]$. That is, $|A_1 \cup A_2 \cup A_3| < k$ for every $A_1, A_2, A_3 \notin \Gamma$.

In the case of PIR, we treat the unauthorized sets of access structure Γ as the sets of servers an adversary may actively corrupt.

Theorem 5.7 (Q_3-PIR is robust)**.** *Any PIR scheme with a Q_3 Access Structure is* Robust*. That is, it is possible to define an alternative* robust decoding *procedure that recovers the queried index bit even if a Byzantine adversary actively controls an unauthorized set of servers and returns malicious responses. The complexity and error of the robust decoder is at most $O(2^k T + 2^{2k})$ and $2^k \cdot \delta$, respectively, where T and δ are the complexity and error of the original decoder.*

The proof is standard and deferred to Section D. For example, by instantiating Corollary 5.6 with t-out-of-$k = 3t + 1$ access structure, we get a PIR that remains actively secure against any byzantine adversary that corrupts up to t servers. Note that the notion of robust decoding allows us to achieve correctness even in the presence of a computationally unbounded active adversary. Further note that this is achieved without giving the client any auxiliary information about the database.

6 Communication-efficient MPC

In this section, we show how to use the PIR protocols from Section 5 to derive succinct MPC protocols for truth tables. Throughout this section, we assume familiarity with standard MPC security definitions (see, e.g., [49, Chapter 7]). We assume a synchronous network in which the parties are connected via point-to-point channels and have access to a broadcast channel. The adversary is always assumed to be rushing, non-adaptive, and computationally bounded. For simplicity, we focus on public-output functionalities that deliver the same output to all parties. (The results can be easily generalized to the more general setting of multi-output functionalities.)

We prove the following theorem that derives short MPC for truth tables based on short PIR in various settings.

Theorem 6.1 (Succinct MPC from short PIR)**.** *Suppose that we have* short PIR*: namely, for every constant $k \geq 2$ and every $\delta > 0$, there exists a k-server fully private PIR with communication complexity of n^δ, querying/decoding complexity of n^δ, and polynomial-time answer algorithm, for databases of size n.*

Then, for every constants $\epsilon > 0$ and $k \geq 2$, every k-party public-output functionality

$$f : [n_1] \times \cdots \times [n_k] \to \{0,1\}^m$$

represented by a truth table with $n = \prod_{i=1}^{k} n_i$ entries, where $m = \mathrm{poly}(n)$, can be securely realized with $\mathrm{poly}(n)$ computational efficiency and communication complexity $O_k(mn^\epsilon)$ in the following settings:

Protocol Π_1: *Passive security against an arbitrary coalition, assuming the existence of OT.*

Protocol Π_2: *Active security with guaranteed output delivery (GOD) assuming an honest majority.*

Protocol Π_3: *Active security with abort against an arbitrary coalition assuming the existence of OT and collision-resistant hash functions.*

The computational and communication complexity of protocols Π_1 and Π_3 grow polynomially with the number of parties k. In protocol Π_2, the complexity is exponential in k. This can be reduced to polynomial in k assuming collision-resistant hash functions (Protocol Π_4).

We instantiate all cryptographic primitives with security parameter $\lambda = n^{\epsilon/2}$, so security is against $\mathrm{poly}(n)$-size adversaries. Since under LPN assumption we have both short-PIR (Corollary 5.3), and OT [4], Theorem 1.4 follows.

Proof overview. Fix constants $k, \epsilon > 0$. Let $f(y_1, \ldots, y_k)$ be a k-party public-output functionality given by an n-size truth table. For $i = 1, \ldots, 4$, we base protocol Π_i on a protocol $\Pi_i^{g_1,g_2}$ in a hybrid model where the parties have access to a pair of ideal functionalities (g_1, g_2) whose circuit complexity is $O_k(mn^{0.5\epsilon})$.

By the seminal completeness results of MPC (e.g., [51, 17, 30, 69]), the functionalities g_1 and g_2 can be securely computed in any of the settings considered in Theorem 6.1, with computational and communication complexity proportional to their circuit size and the security parameter λ. (In the dishonest-majority setting, this also requires OT.) By standard composition theorems (e.g., [49, Thms. 7.3.3, 7.4.3]), we can replace the calls to the ideal functionalities in $\Pi_i^{g_1,g_2}$ with the corresponding protocols while preserving security.

Taking $\lambda = O(n^{\epsilon/2})$, we obtain protocols with computational complexity $\mathrm{poly}(n)$ and communication complexity $O_k(mn^\epsilon)$, as required.

The protocols $\Pi_1^{g_1,g_2}, \ldots, \Pi_4^{g_1,g_2}$ are described and analyzed in the following sections.

6.1 Passive security for arbitrary coalitions

The protocol $\Pi_1^{g_1,g_2}$. Let $(\textsc{Query}, \textsc{Ans}, \textsc{Dec})$ be a fully private k-server PIR scheme in which the functions $\textsc{Query}$ and $\textsc{Dec}$ have circuit complexity $O_k(mn^{0.5\epsilon})$ for a database of size n consisting of m-bit entries. Such a PIR scheme exists by assumption.[17]

We view the truth table of f as a public database $F \in (\{0,1\}^m)^n$, and a k-party input $y = (y_1, \ldots, y_k)$ as a target index; namely, $f(y) = F[y] \in \{0,1\}^m$. The ideal functionalities g_1 and g_2 are based on the PIR algorithms $\textsc{Query}(1^n, \cdot)$ and $\textsc{Dec}(1^n, \cdot)$. (We omit the unary input from now on.) Specifically, the protocol $\Pi_1^{g_1,g_2}$ proceeds as follows:

[17] The assumption is stated for databases with single-bit entries. The extension to m-bit entries follows via a standard reduction: view the database as m separate single-bit databases, one for each bit position, and let the servers compute their answers independently for each database. This increases both the computation and communication by a factor of at most m.

1. The parties invoke the ideal functionality QUERY with inputs $y = (y_1, \ldots, y_k)$ and local randomness $r_1, \ldots, r_k \xleftarrow{R} \{0,1\}^{\rho(n)}$, where $\rho(n)$ denotes the randomness complexity of the query algorithm. The functionality computes
$$(q_1, \ldots, q_k) = \text{QUERY}(y; r = \sum_i r_i),$$
and returns the query q_j to party P_j. (Here addition denotes bitwise XOR.)

2. Each party P_j locally computes its PIR answer $\mathbf{ans}_j = \text{ANS}(F, q_j)$. (Here and throughout this section we implicitly assume that the query q_j contains the "label" j of the server and so the answering algorithm does not need the index j as an additional input.)

3. The parties invoke the ideal functionality DEC with inputs $y = (y_1, \ldots, y_k)$, randomness $r = \sum_i r_i$, and answers $\mathbf{ans} = (\mathbf{ans}_1, \ldots, \mathbf{ans}_k)$, and receive the output bit $F[y] = \text{DEC}(y, r = \sum_i r_i, \mathbf{ans})$.

Lemma 6.2. *The protocol $\Pi_1^{g_1,g_2}$ securely computes the functionality f in the (g_1, g_2)-hybrid model with passive security against an arbitrary coalition.*

Lemma 6.2, whose proof is deferred to Section E.1, concludes the proof of Protocol Π_1 in Theorem 6.1. □

6.2 Active Security with GOD assuming Honest Majority

In order to obtain a protocol with GOD for f, it suffices to obtain a protocol with fairness and identifiable abort. That is, the adversary may cause the protocol to abort for all parties (including the corrupted ones), but upon abort, all honest parties agree on the identity of at least one corrupted party.

Formally, this is captured by a GOD protocol for the functionality f_{fairID}. The functionality receives inputs $y = (y_1, \ldots, y_k)$ from the parties. In addition, the adversary may send a special abort command consisting of $\perp$ together with a subset $T' \subseteq T$ of the corrupted parties. Upon receiving such an abort command, the functionality broadcasts $(\perp, T')$ to all parties. Otherwise, it outputs $f(y)$.

By a standard player-emulation technique (see, e.g., [53]), the problem of securely computing f with GOD can be reduced to the problem of securely computing f_{fairID} with GOD.

Proposition 6.3. *For every function f, suppose there exists a protocol Π' that securely computes f_{fairID} with active security and guaranteed output delivery in the honest-majority setting. Then there exists a protocol Π that securely computes f with active security and guaranteed output delivery in the honest-majority setting.*

Moreover, the computational and communication complexity of Π are at most a factor of $k/2$ larger than those of Π', where k denotes the number of parties.

Proof sketch. The parties repeatedly invoke Π'. Whenever an execution outputs $(\perp, T')$, the parties eliminate all parties in T'. Since the adversary corrupts at most $t < k/2$ parties, each abort eliminates at least one corrupted party, and therefore the number of iterations is at most t. Consequently, the overall computational and communication complexity increase by at most a factor of $t < k/2$. The construction of the simulator is straightforward. □

We construct a protocol for f_{fairID}. Assume that $k = 2t + 1$, and let $K = \binom{k}{t}$. The idea is to employ a K-server fully private PIR scheme in which each server S_B is associated with a t-subset $B \subset [k]$. Each party P_i emulates all servers S_B for which $i \notin B$. Intuitively, an adversary corrupting a set $T \subset [k]$ of at most t parties does not observe the view of the server S_T. On the other hand, since there is an honest

majority, every server S_B is emulated by at least one honest party. Using standard techniques, this suffices to guarantee output delivery. We now give the details.

Let $(\text{QUERY}, \text{ANS}, \text{DEC})$ be a fully private K-server PIR scheme in which the functions QUERY and DEC have circuit complexity $\ell = O_K(mn^{0.5\epsilon})$ for a database of size n consisting of m-bit entries. Such a PIR scheme exists by assumption. (See Footnote 17.) We also employ the following standard notion of t-robust secret sharing.

Robust secret sharing. A t-robust k-party secret-sharing scheme is a pair of efficient algorithms SHARE and REC. Given a message M and randomness R, the randomized sharing algorithm outputs shares

$$(S_1, \ldots, S_k) = \text{SHARE}(M; R).$$

The scheme satisfies *t-privacy*: for every pair of messages M, M' and every $T \subseteq [k]$ of size at most t, the distributions of

$$S_T = (S_i)_{i \in T} \qquad \text{and} \qquad S'_T = (S'_i)_{i \in T}$$

are identical, where $(S_i)_{i \in [k]} = \text{SHARE}(M; R)$ and $(S'_i)_{i \in [k]} = \text{SHARE}(M'; R')$ for uniformly sampled randomness R, R'. The scheme further satisfies *t-robustness*: if a share vector $\hat{S} = (\hat{S}_i)_{i \in [k]}$ differs from a valid sharing of some message M in at most t coordinates, then $\text{REC}(\hat{S}) = M$. (In particular, such an M is unique.) If no such message exists, then $\text{REC}(\hat{S}) = \perp$. We assume that sharing and reconstruction of an ℓ-bit message require time $\tilde{O}(\ell, k)$. In the honest-majority setting ($k = 2t + 1$), these properties are achieved by Shamir's polynomial-based secret-sharing scheme [70].

The protocol $\Pi_2^{g_1, g_2}$. The protocol for f_{fairID} proceeds as follows:

1. The parties invoke the ideal randomized functionality g_1 with inputs $y = (y_1, \ldots, y_k)$ and local randomness $r_1, \ldots, r_k \xleftarrow{R} \{0,1\}^{\rho(n)}$ where $\rho(n)$ is the randomness complexity of the query algorithm. The functionality sets the PIR randomness $r = \sum_i r_i$, computes the queries $(q_S)_{S \in \binom{[k]}{t}} = \text{QUERY}(y; r)$, and secret-shares both the queries and the input/randomness:

$$(Q_{j,S})_{j \in [k]} = \text{SHARE}(q_S), \quad \forall S \in \binom{[k]}{t}, \qquad (C_{j,i})_{j \in [k]} = \text{SHARE}(y_i, r_i) \quad \forall i \in [k].$$

   The functionality delivers to P_j its shares together with all queries q_S for which $j \notin S$, i.e.,

$$(Q_{j,S})_{S \in \binom{[k]}{t}}, \quad (C_{j,i})_{i \in [k]}, \quad \text{and} \quad (q_S)_{S : j \notin S}.$$

2. Each party P_j locally computes the PIR answers $\mathbf{ans}_{j,S} = \text{ANS}(F, q_S)$ for all S such that $j \notin S$.

3. The parties invoke the ideal functionality g_2 that takes from party P_j the input

$$(\mathbf{ans}_{j,S})_{S : j \notin S}, \qquad (C_{j,i})_{i \in [k]},$$

   and proceeds as follows. If, for every $S \in \binom{[k]}{t}$, all values $(\mathbf{ans}_{j,S})_{j \notin S}$ agree on a common value $\mathbf{ans}_S$, the functionality reconstructs $(y_i, r_i) = \text{REC}((C_{j,i})_{j \in [k]})$ for every $i \in [k]$, sets $y = (y_1, \ldots, y_k)$, $r = \sum_i r_i$, and $\mathbf{ans} = (\mathbf{ans}_S)_{S \in \binom{[k]}{t}}$, and broadcasts $z = \text{DEC}(y, r, \mathbf{ans})$.

Otherwise, the functionality broadcasts the lexicographically first conflict tuple

$$(i, i', S, \mathbf{ans}_{i,S}, \mathbf{ans}_{i',S})$$

for which $\mathbf{ans}_{i,S} \neq \mathbf{ans}_{i',S}$.

4. If the parties receive an output value z, they terminate and output z. Otherwise, each party P_j broadcasts its share $Q_{j,S}$ of the conflicting query. The parties reconstruct q_S via robust recovery and compute $\mathbf{ans}_S = \text{ANS}(F, q_S)$. They then output $(\perp, \{i\})$ if $\mathbf{ans}_{i,S} \neq \mathbf{ans}_S$, and otherwise output $(\perp, \{i'\})$.

Lemma 6.4 (security of $\Pi_2^{g_1,g_2}$)**.** *The protocol $\Pi_2^{g_1,g_2}$ securely computes the k-party functionality f_{fairID} in the (g_1, g_2)-hybrid model with active security and guaranteed output delivery, assuming an honest majority.*

The proof is deferred to Section E.2. Together with Proposition 6.3, this concludes the proof of Protocol Π_2 in Theorem 6.1. □

6.3 Active Full Security with abort

The protocol $\Pi_3^{g_1,g_2}$ follows the general structure of the first protocol $\Pi_1^{g_1,g_2}$. To cope with an active adversary, we apply several standard modifications similar in spirit to the well-known GMW compiler [51]. Note that unlike the previous section here we consider the standard Security with abort model that allows the adversary to abort the honest parties after learning the output of the function.

As before, let $(\text{QUERY}, \text{ANS}, \text{DEC})$ be a fully private k-server PIR scheme in which the functions QUERY and DEC have circuit complexity $\ell = O_k(mn^{0.5\epsilon})$ for a database of size n consisting of m-bit entries. Such a PIR scheme exists by assumption. (See Footnote 17.)

Additional cryptographic tools. We also employ the following tools.

1. An additive k-out-of-k secret-sharing scheme $(\text{SHARE}, \text{REC})$.

2. A non-interactive commitment scheme (NICOM) with computational hiding and statistical binding for *honestly generated* commitments. A NICOM is a randomized algorithm COM that, given a message M and randomness R, outputs a commitment $C = \text{COM}(M; R)$. We refer to R as the opening of the commitment. The scheme is computationally hiding: for every pair of messages M, M', the commitments

$$C = \text{COM}(M; R) \qquad \text{and} \qquad C' = \text{COM}(M'; R')$$

are computationally indistinguishable, where R and R' are chosen uniformly at random.

In addition, the scheme is statistically binding for honestly generated commitments: for every message M and every honestly generated commitment $C = \text{COM}(M; R)$, except with negligible probability over the choice of R, there do not exist $M' \neq M$ and R' such that $C = \text{COM}(M'; R')$.

These properties are satisfied by Naor's commitment scheme [68], which can be based on any one-way function and therefore based on collision-resistant hash functions.

3. A digital signature scheme $(\text{KG}, \text{SIGN}, \text{VER})$ whose existence can be based on any one-way function (see [49]) and therefore also on collision-resistant hash functions.

4. A zero-knowledge argument system for proving the satisfiability of an S-size circuit with communication complexity $\lambda \operatorname{polylog}(S)$, where λ is the security parameter. We assume, for convenience, that the system has perfect completeness. Such a system can be constructed from collision-resistant hash functions by the result of Kilian [60].

We further assume that the length of a commitment or signature is linear in the message length and the security parameter λ. (Standard constructions satisfy this property.) Recall also that the security parameter is chosen to be $\lambda = O(n^{0.5\epsilon})$.

The protocol $\Pi_3^{g_1,g_2}$. The protocol proceeds as follows.

1. The parties invoke the ideal randomized functionality g_1 that takes from each party P_i its input y_i together with local PIR randomness $r_i \xleftarrow{R} \{0,1\}^{\rho(n)}$, where $\rho(n)$ denotes the randomness complexity of the query algorithm QUERY. The functionality then computes

$$(q_1, \ldots, q_k) = \text{QUERY}(y; r), \qquad \text{where} \qquad y = (y_i)_{i \in [k]} \quad \text{and} \quad r = \sum_i r_i,$$

   and returns the query q_j to party P_j. In addition, for every $i \in [k]$, the functionality samples commitment randomness o_i, computes the commitment

$$Q_i = \text{COM}(y_i, r_i, q_i; o_i),$$

   broadcasts the commitment vector $Q = (Q_1, \ldots, Q_k)$ to all parties, and delivers the opening o_i to party P_i.

2. Each party P_j locally computes its PIR answer $\mathbf{ans}_j = \text{ANS}(F, q_j)$.

3. The parties invoke the ideal randomized functionality g_2 that takes from each party P_i the tuple $(y_i, r_i, \mathbf{ans}_i)$. The functionality computes

$$z = \text{DEC}(y, r, \mathbf{ans}), \qquad \text{where} \qquad \mathbf{ans} = (\mathbf{ans}_i)_{i \in [k]},$$

   shares z as $(z_1, \ldots, z_k) = \text{SHARE}(z)$, and returns the share z_i to party P_i.

   In addition, the functionality samples a signing key and verification key pair $(\mathsf{sk}, \mathsf{vk})$. For every $i \in [k]$, it samples commitment randomness a_i and computes

$$A_i = \text{COM}(y_i, r_i, \mathbf{ans}_i; a_i), \qquad \zeta_i = \text{SIGN}_{\mathsf{sk}}(i, z_i).$$

   The functionality broadcasts the verification key vk and the commitment vector $A = (A_i)_{i \in [k]}$ to all parties, and delivers to P_i the tuple (z_i, ζ_i, a_i).

4. For every ordered pair $i \neq j$ (processed in lexicographic order), party P_i proves to party P_j in zero knowledge that it knows values $(y_i, r_i, q_i, \mathbf{ans}_i)$ and openings (o_i, a_i) satisfying

$$Q_i = \text{COM}(y_i, r_i, q_i; o_i), \qquad A_i = \text{COM}(y_i, r_i, \mathbf{ans}_i; a_i), \qquad \mathbf{ans}_i = \text{ANS}(F, q_i). \tag{3}$$

   Party P_j broadcasts the resulting verification bit $b_{i,j}$. For notational convenience, set $b_{i,i} = 1$ for every $i \in [k]$.

5. Let $b = \wedge_{i,j} b_{i,j}$. If $b = 0$, the parties abort. Otherwise:

   (a) Each party P_i broadcasts (z_i, ζ_i).

   (b) If all signatures verify, i.e., $\text{VER}_{\mathsf{vk}}((i, z_i), \zeta_i) = 1$ for all $i \in [k]$ then each party reconstructs $z = \text{REC}(z_1, \ldots, z_k)$. Otherwise, the party aborts.

Lemma 6.5 (Active security with abort)**.** *The protocol $\Pi_3^{g_1,g_2}$ securely computes the functionality f in the (g_1, g_2)-hybrid model with active security and abort against an arbitrary coalition.*

The proof is deferred to Section E.3. This concludes the proof of Protocol Π_3 in Theorem 6.1. □

6.4 Active Security with GOD assuming Honest Majority with $\text{poly}(k)$ Overhead

Finally, we note that by slightly modifying the previous protocol, it is possible to achieve GOD against an active adversary assuming honest majority. Unlike protocol Π_2, that achieved this task with exponential overhead in the number of parties, the current protocol Π_4 has only polynomial overhead in k.

By Proposition 6.3, it suffices to construct a protocol $\Pi_4^{g_1,g_2}$ for f_{fairID}. The protocol is identical to $\Pi_3^{g_1,g_2}$ except for the following changes.

Shamir's secret-sharing. The underlying secret-sharing $(\text{SHARE}, \text{REC})$ is based on Shamir's secret sharing scheme with privacy threshold t. (Recall that $k = 2t + 1$.) The correctness property guarantees that for every secret s, vector of shares $(s_i)_{i\in[k]}$ in the support of $\text{SHARE}(s)$ and set A of size at least $t + 1$, the reconstruction algorithm satisfies

$$\text{REC}(A, (s_i)_{i\in A}) = s.$$

Just like in the case of additive secret sharing the scheme is linear. We can therefore sample random partial sharing $(s_i)_{i\in T}$ for a subset $|T| \leq t$, and later given a "target secret" s, efficiently sample $(s_i)_{i\in\bar{T}}$ conditioned on $(s_i)_{i\in[k]} \overset{R}{\leftarrow} \text{SHARE}(s)$.

The final step. The last Step of the protocol is replaced as follows.

5. For every $i \in [k]$, let $b_i = \text{MAJ}(b_{i,1}, \ldots, b_{i,k})$. Let $b = \wedge_i b_i$.

   - If $b = 0$, the parties abort the protocol and point to the parties in $B = \{i : b_i = 0\}$ as corrupted.
   - Else, (if $b = 1$) each party P_j releases its share z_j and the signature ζ_j. Let $S = (j : \text{VER}_{\mathsf{vk}}((j, z_j), \zeta_j) = 1)$, each party then uses the secret-sharing recovery algorithm to compute $z = \text{REC}(S, (z_i)_{i\in S})$ and outputs the result.

Lemma 6.6 (Security of $\Pi_4^{g_1,g_2}$)**.** *The protocol $\Pi_4^{g_1,g_2}$ securely computes the functionality f_{fairID} in the (g_1, g_2)-hybrid model with active security and GOD assuming honest majority.*

The proof is deferred to Section E.3. Together with Proposition 6.3, this concludes the proof of Protocol Π_4 in Theorem 6.1. □

Acknowledgment. We thank Amos Beimel for pointing out the PIR-to-MPC reduction implicit in [13]. We also thank Yuval Ishai for helpful discussions.

References

[1] Damiano Abram, Giulio Malavolta, and Lawrence Roy. Slightly sublinear trapdoor hash functions and PIR from low-noise LPN. In Benny Applebaum and Huijia Rachel Lin, editors, *Theory of Cryptography - 23rd International Conference, TCC 2025, Aarhus, Denmark, December 1-5, 2025, Proceedings, Part I*, volume 16268 of *Lecture Notes in Computer Science*, pages 280–300. Springer, 2025.

[2] Damiano Abram, Lawrence Roy, and Peter Scholl. Succinct homomorphic secret sharing. In Marc Joye and Gregor Leander, editors, *Advances in Cryptology - EUROCRYPT 2024 - 43rd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zurich, Switzerland, May 26-30, 2024, Proceedings, Part VI*, volume 14656 of *Lecture Notes in Computer Science*, pages 301–330. Springer, 2024.

[3] Amit Agarwal, Elette Boyle, Niv Gilboa, Yuval Ishai, Mahimna Kelkar, and Yiping Ma. Compressing unit-vector correlations via sparse pseudorandom generators. In Leonid Reyzin and Douglas Stebila, editors, *Advances in Cryptology - CRYPTO 2024 - 44th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2024, Proceedings, Part VIII*, volume 14927 of *Lecture Notes in Computer Science*, pages 346–383. Springer, 2024.

[4] Michael Alekhnovich. More on average case vs approximation complexity. In *44th Symposium on Foundations of Computer Science (FOCS 2003), Cambridge, MA, USA, October 11-14, 2003, Proceedings*, pages 298–307. IEEE Computer Society, 2003.

[5] Bar Alon, Amos Beimel, and Or Lasri. Simplified pir and cds protocols and improved linear secret-sharing schemes. In *Theory of Cryptography Conference*, pages 365–398. Springer, 2025.

[6] Benny Applebaum, Jonathan Avron, and Christina Brzuska. Arithmetic cryptography: Extended abstract. In *Proceedings of the 6th Innovations in Theoretical Computer Science Conference (ITCS 2015)*, pages 143–151. ACM, 2015.

[7] Benny Applebaum and Aarushi Goel. On actively-secure elementary MPC reductions. In Kobbi Nissim and Brent Waters, editors, *Theory of Cryptography - 19th International Conference, TCC 2021, Raleigh, NC, USA, November 8-11, 2021, Proceedings, Part I*, volume 13042 of *Lecture Notes in Computer Science*, pages 717–749. Springer, 2021.

[8] Benny Applebaum, Yuval Ishai, and Eyal Kushilevitz. Cryptography with constant input locality. In Alfred Menezes, editor, *Advances in Cryptology - CRYPTO 2007, 27th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2007, Proceedings*, volume 4622 of *Lecture Notes in Computer Science*, pages 92–110. Springer, 2007.

[9] Omer Barkol, Yuval Ishai, and Enav Weinreb. On locally decodable codes, self-correctable codes, and t-private PIR. *Algorithmica*, 58(4):831–859, 2010.

[10] David A. Mix Barrington, Richard Beigel, and Steven Rudich. Representing boolean functions as polynomials modulo composite numbers (extended abstract). In S. Rao Kosaraju, Mike Fellows, Avi Wigderson, and John A. Ellis, editors, *Proceedings of the 24th Annual ACM Symposium on Theory of Computing, May 4-6, 1992, Victoria, British Columbia, Canada*, pages 455–461. ACM, 1992.

[11] Amos Beimel, Oriol Farràs, and Or Lasri. Improved polynomial secret-sharing schemes. In *Theory of Cryptography Conference*, pages 374–405. Springer, 2023.

[12] Amos Beimel and Yuval Ishai. Information-theoretic private information retrieval: A unified construction. In Fernando Orejas, Paul G. Spirakis, and Jan van Leeuwen, editors, *Automata, Languages and Programming, 28th International Colloquium, ICALP 2001, Crete, Greece, July 8-12, 2001, Proceedings*, volume 2076 of *Lecture Notes in Computer Science*, pages 912–926. Springer, 2001.

[13] Amos Beimel, Yuval Ishai, Ranjit Kumaresan, and Eyal Kushilevitz. On the cryptographic complexity of the worst functions. In Yehuda Lindell, editor, *Theory of Cryptography - 11th Theory of Cryptography Conference, TCC 2014, San Diego, CA, USA, February 24-26, 2014. Proceedings*, volume 8349 of *Lecture Notes in Computer Science*, pages 317–342. Springer, 2014.

[14] Amos Beimel, Yuval Ishai, and Eyal Kushilevitz. General constructions for information-theoretic private information retrieval. *J. Comput. Syst. Sci.*, 71(2):213–247, 2005.

[15] Amos Beimel, Yuval Ishai, and Tal Malkin. Reducing the servers' computation in private information retrieval: PIR with preprocessing. *J. Cryptol.*, 17(2):125–151, 2004.

[16] Amos Beimel and Yoav Stahl. Robust information-theoretic private information retrieval. *J. Cryptol.*, 20(3):295–321, 2007.

[17] Michael Ben-Or, Shafi Goldwasser, and Avi Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation. In Jan Simon, editor, *Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing (STOC 1988)*, pages 1–10. ACM, 1988.

[18] Josh Benaloh and Jerry Leichter. Generalized secret sharing and monotone functions. In *CRYPTO '88*, volume 403 of *LNCS*, pages 27–35. Springer, 1990.

[19] Avrim Blum, Merrick L. Furst, Michael J. Kearns, and Richard J. Lipton. Cryptographic primitives based on hard learning problems. In Douglas R. Stinson, editor, *Advances in Cryptology - CRYPTO '93, 13th Annual International Cryptology Conference, Santa Barbara, California, USA, August 22-26, 1993, Proceedings*, volume 773 of *Lecture Notes in Computer Science*, pages 278–291. Springer, 1993.

[20] Elette Boyle, Geoffroy Couteau, Niv Gilboa, and Yuval Ishai. Compressing vector OLE. In David Lie, Mohammad Mannan, Michael Backes, and XiaoFeng Wang, editors, *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security (CCS 2018)*, pages 896–912. ACM, 2018.

[21] Elette Boyle, Geoffroy Couteau, Niv Gilboa, Yuval Ishai, Lisa Kohl, and Peter Scholl. Efficient pseudorandom correlation generators: Silent OT extension and more. In Alexandra Boldyreva and Daniele Micciancio, editors, *Advances in Cryptology - CRYPTO 2019 - 39th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2019, Proceedings, Part III*, volume 11694 of *Lecture Notes in Computer Science*, pages 489–518. Springer, 2019.

[22] Elette Boyle, Geoffroy Couteau, and Pierre Meyer. Sublinear-communication secure multiparty computation does not require FHE. In Carmit Hazay and Martijn Stam, editors, *Advances in Cryptology - EUROCRYPT 2023 - 42nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Lyon, France, April 23-27, 2023, Proceedings, Part II*, volume 14005 of *Lecture Notes in Computer Science*, pages 159–189. Springer, 2023.

[23] Elette Boyle, Niv Gilboa, Matan Hamilis, Yuval Ishai, and Ariel Nof. Dishonest-majority secure computation via pir-authenticated multiplication triples. *IACR Cryptol. ePrint Arch.*, 2026:257, 2026.

[24] Elette Boyle, Niv Gilboa, and Yuval Ishai. Function secret sharing. In Elisabeth Oswald and Marc Fischlin, editors, *Advances in Cryptology - EUROCRYPT 2015 - 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Sofia, Bulgaria, April 26-30, 2015, Proceedings, Part II*, volume 9057 of *Lecture Notes in Computer Science*, pages 337–367. Springer, 2015.

[25] Elette Boyle, Niv Gilboa, and Yuval Ishai. Breaking the circuit size barrier for secure computation under DDH. In Matthew Robshaw and Jonathan Katz, editors, *Advances in Cryptology - CRYPTO 2016 - 36th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2016, Proceedings, Part I*, volume 9814 of *Lecture Notes in Computer Science*, pages 509–539. Springer, 2016.

[26] Elette Boyle, Niv Gilboa, and Yuval Ishai. Function secret sharing: Improvements and extensions. In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, pages 1292–1303, 2016.

[27] Elette Boyle, Niv Gilboa, Yuval Ishai, and Victor I. Kolobov. Information-theoretic distributed point functions. In Dana Dachman-Soled, editor, *3rd Conference on Information-Theoretic Cryptography, ITC 2022, Cambridge, MA, USA, July 5-7, 2022*, volume 230 of *LIPIcs*, pages 17:1–17:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022.

[28] Elette Boyle, Yuval Ishai, Rafael Pass, and Mary Wootters. Can we access a database both locally and privately? In Yael Kalai and Leonid Reyzin, editors, *Theory of Cryptography - 15th International Conference, TCC 2017, Baltimore, MD, USA, November 12-15, 2017, Proceedings, Part II*, volume 10678 of *Lecture Notes in Computer Science*, pages 662–693. Springer, 2017.

[29] Ran Canetti, Justin Holmgren, and Silas Richelson. Towards doubly efficient private information retrieval. *IACR Cryptol. ePrint Arch.*, page 568, 2017.

[30] David Chaum, Claude Crépeau, and Ivan Damgård. Multiparty unconditionally secure protocols. In Carl Pomerance, editor, *Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing (STOC 1988)*, pages 11–19. ACM, 1988.

[31] Caicai Chen, Yuval Ishai, Tamer Mour, and Alon Rosen. Secret-key PIR from random linear codes. Cryptology ePrint Archive, Paper 2025/646, 2025.

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

[33] Ilaria Chillotti, Emmanuela Orsini, Peter Scholl, and Barry Van Leeuwen. Scooby: Improved multi-party homomorphic secret sharing based on FHE. *Inf. Comput.*, 297:105133, 2024.

[34] Benny Chor and Niv Gilboa. Computationally private information retrieval (extended abstract). In Frank Thomson Leighton and Peter W. Shor, editors, *Proceedings of the Twenty-Ninth Annual ACM Symposium on the Theory of Computing, El Paso, Texas, USA, May 4-6, 1997*, pages 304–313. ACM, 1997.

[35] Benny Chor, Eyal Kushilevitz, Oded Goldreich, and Madhu Sudan. Private information retrieval. *J. ACM*, 45(6):965–981, 1998.

[36] Gil Cohen, Ivan Bjerre Damgård, Yuval Ishai, Jonas Kölker, Peter Bro Miltersen, Ran Raz, and Ron D. Rothblum. Efficient multiparty protocols via log-depth threshold formulae - (extended abstract). In Ran Canetti and Juan A. Garay, editors, *Advances in Cryptology - CRYPTO 2013 - 33rd Annual Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2013. Proceedings, Part II*, volume 8043 of *Lecture Notes in Computer Science*, pages 185–202. Springer, 2013.

[37] Henry Corrigan-Gibbs. Private information retrieval at 30: Achievements, disappointments, and open problems. Invited talk, Workshop on Secure Computation, August 6, 2025, 2025. `https://simons.berkeley.edu/talks/henry-corrigan-gibbs-mit-2025-08-06`.

[38] Henry Corrigan-Gibbs and Dmitry Kogan. Private information retrieval with sublinear online time. In Anne Canteaut and Yuval Ishai, editors, *Advances in Cryptology - EUROCRYPT 2020 - 39th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, May 10-14, 2020, Proceedings, Part I*, volume 12105 of *Lecture Notes in Computer Science*, pages 44–75. Springer, 2020.

[39] Geoffroy Couteau and Naman Kumar. 10-party sublinear secure computation from standard assumptions. *IACR Cryptol. ePrint Arch.*, page 269, 2025.

[40] Geoffroy Couteau, Naman Kumar, and Xiaxi Ye. Multiparty homomorphic secret sharing and more from LPN and MQ. In Benny Applebaum and Huijia Rachel Lin, editors, *Theory of Cryptography - 23rd International Conference, TCC 2025, Aarhus, Denmark, December 1-5, 2025, Proceedings, Part II*, volume 16269 of *Lecture Notes in Computer Science*, pages 498–530. Springer, 2025.

[41] Ronald Cramer, Ivan Damgård, and Jesper Buus Nielsen. *Secure Multiparty Computation and Secret Sharing*. Cambridge University Press, 2015.

[42] Quang Dao, Yuval Ishai, Aayush Jain, and Huijia Lin. Multi-party homomorphic secret sharing and sublinear MPC from sparse LPN. In Helena Handschuh and Anna Lysyanskaya, editors, *Advances in Cryptology - CRYPTO 2023 - 43rd Annual International Cryptology Conference, CRYPTO 2023, Santa Barbara, CA, USA, August 20-24, 2023, Proceedings, Part II*, volume 14082 of *Lecture Notes in Computer Science*, pages 315–348. Springer, 2023.

[43] Nico Döttling. Low noise LPN: KDM secure public key encryption and sample amplification. In Jonathan Katz, editor, *Public-Key Cryptography - PKC 2015 - 18th IACR International Conference on Practice and Theory in Public-Key Cryptography, Gaithersburg, MD, USA, March 30 - April 1, 2015, Proceedings*, volume 9020 of *Lecture Notes in Computer Science*, pages 604–626. Springer, 2015.

[44] Nico Döttling, Jörn Müller-Quade, and Anderson C. A. Nascimento. IND-CCA secure cryptography based on a variant of the LPN problem. In Xiaoyun Wang and Kazue Sako, editors, *Advances in Cryptology - ASIACRYPT 2012 - 18th International Conference on the Theory and Application of Cryptology and Information Security, Beijing, China, December 2-6, 2012, Proceedings*, volume 7658 of *Lecture Notes in Computer Science*, pages 485–503. Springer, 2012.

[45] Zeev Dvir and Sivakanth Gopi. 2-server PIR with sub-polynomial communication. In Rocco A. Servedio and Ronitt Rubinfeld, editors, *Proceedings of the Forty-Seventh Annual ACM on Symposium*

on Theory of Computing, STOC 2015, Portland, OR, USA, June 14-17, 2015, pages 577–584. ACM, 2015.

[46] Ashrujit Ghoshal, Baitian Li, Yaohua Ma, Chenxin Dai, and Elaine Shi. Scalable multi-server private information retrieval. In Benny Applebaum and Huijia (Rachel) Lin, editors, *Theory of Cryptography - 23rd International Conference, TCC 2025, Aarhus, Denmark, December 1-5, 2025, Proceedings, Part IV*, volume 16271 of *Lecture Notes in Computer Science*, pages 582–610. Springer, 2025.

[47] Niv Gilboa and Yuval Ishai. Distributed point functions and their applications. In Phong Q. Nguyen and Elisabeth Oswald, editors, *Advances in Cryptology - EUROCRYPT 2014 - 33rd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Copenhagen, Denmark, May 11-15, 2014. Proceedings*, volume 8441 of *Lecture Notes in Computer Science*, pages 640–658. Springer, 2014.

[48] Aarushi Goel, Mingyuan Wang, and Zhiheng Wang. Multiparty distributed point functions. In Yael Tauman Kalai and Seny F. Kamara, editors, *Advances in Cryptology - CRYPTO 2025 - 45th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2025, Proceedings, Part IV*, volume 16003 of *Lecture Notes in Computer Science*, pages 140–173. Springer, 2025.

[49] Oded Goldreich. *The Foundations of Cryptography - Volume 2: Basic Applications*. Cambridge University Press, 2004.

[50] Oded Goldreich, Hugo Krawczyk, and Michael Luby. On the existence of pseudorandom generators. *SIAM J. Comput.*, 22(6):1163–1175, 1993.

[51] Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game, or a completeness theorem for protocols with honest majority. In Oded Goldreich, editor, *Providing Sound Foundations for Cryptography: On the Work of Shafi Goldwasser and Silvio Micali*, pages 307–328. ACM, 2019.

[52] Martin Hirt and Ueli M. Maurer. Player simulation and general adversary structures in perfect multiparty computation. *J. Cryptol.*, 13(1):31–60, 2000.

[53] Martin Hirt and Jesper Buus Nielsen. Upper bounds on the communication complexity of optimally resilient cryptographic multiparty computation. In Bimal K. Roy, editor, *Advances in Cryptology - ASIACRYPT 2005, 11th International Conference on the Theory and Application of Cryptology and Information Security, Chennai, India, December 4-8, 2005, Proceedings*, volume 3788 of *Lecture Notes in Computer Science*, pages 79–99. Springer, 2005.

[54] Yuval Ishai and Eyal Kushilevitz. Improved upper bounds on information-theoretic private information retrieval (extended abstract). In Jeffrey Scott Vitter, Lawrence L. Larmore, and Frank Thomson Leighton, editors, *Proceedings of the Thirty-First Annual ACM Symposium on Theory of Computing, May 1-4, 1999, Atlanta, Georgia, USA*, pages 79–88. ACM, 1999.

[55] Yuval Ishai and Eyal Kushilevitz. On the hardness of information-theoretic multiparty computation. In Christian Cachin and Jan Camenisch, editors, *Advances in Cryptology - EUROCRYPT 2004, International Conference on the Theory and Applications of Cryptographic Techniques, Interlaken, Switzerland, May 2-6, 2004, Proceedings*, volume 3027 of *Lecture Notes in Computer Science*, pages 439–455. Springer, 2004.

[56] Yuval Ishai, Manoj Prabhakaran, and Amit Sahai. Secure arithmetic computation with no honest majority. In Omer Reingold, editor, *Theory of Cryptography, 6th Theory of Cryptography Conference, TCC 2009, San Francisco, CA, USA, March 15-17, 2009, Proceedings*, volume 5444 of *Lecture Notes in Computer Science*, pages 294–314. Springer, 2009.

[57] Mitsuru Ito, Akira Saito, and Takao Nishizeki. Secret sharing schemes realizing general access structure. In *Globecom 87*, pages 99–102. IEEE, 1987. Journal version: Multiple assignment scheme for sharing secret. *J. Cryptol.*, 6(1):15-20, 1993.

[58] Aayush Jain, Huijia Lin, and Amit Sahai. Indistinguishability obfuscation from well-founded assumptions. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing (STOC 2021)*, pages 60–73. ACM, 2021.

[59] Aayush Jain, Huijia Lin, and Amit Sahai. Indistinguishability obfuscation from LPN over $\mathbb{F}_p$, DLIN, and PRGs in NC^0. In Orr Dunkelman and Stefan Dziembowski, editors, *Advances in Cryptology - EUROCRYPT 2022 - 41st Annual International Conference on the Theory and Applications of Cryptographic Techniques, Trondheim, Norway, May 30 - June 3, 2022, Proceedings, Part I*, volume 13275 of *Lecture Notes in Computer Science*, pages 670–699. Springer, 2022.

[60] Joe Kilian. A note on efficient zero-knowledge proofs and arguments. In John Simon, editor, *Proceedings of the Twenty-Fourth Annual ACM Symposium on Theory of Computing (STOC 1992)*, pages 723–732. ACM, 1992.

[61] Eike Kiltz, Daniel Masny, and Krzysztof Pietrzak. Simple chosen-ciphertext security from low-noise LPN. In Hugo Krawczyk, editor, *Public-Key Cryptography - PKC 2014 - 17th International Conference on Practice and Theory in Public-Key Cryptography, Buenos Aires, Argentina, March 26-28, 2014, Proceedings*, volume 8383 of *Lecture Notes in Computer Science*, pages 1–18. Springer, 2014.

[62] Eyal Kushilevitz and Rafail Ostrovsky. Replication is NOT needed: SINGLE database, computationally-private information retrieval. In *38th Annual Symposium on Foundations of Computer Science, FOCS 1997, Miami Beach, Florida, USA, October 19-22, 1997*, pages 364–373. IEEE Computer Society, 1997.

[63] Samuel Kutin. Constructing large set systems with given intersection sizes modulo composite numbers. *Comb. Probab. Comput.*, 11(5):475–486, 2002.

[64] Arthur Lazzaretti, Zeyu Liu, Ben Fisch, Peihan Miao, and Charalampos Papamanthou. Multi-server doubly efficient PIR in the classical model and beyond. In Benny Applebaum and Huijia (Rachel) Lin, editors, *Theory of Cryptography - 23rd International Conference, TCC 2025, Aarhus, Denmark, December 1-5, 2025, Proceedings, Part IV*, volume 16271 of *Lecture Notes in Computer Science*, pages 611–641. Springer, 2025.

[65] Wei-Kai Lin, Ethan Mook, and Daniel Wichs. Doubly efficient private information retrieval and fully homomorphic RAM computation from ring LWE. In Barna Saha and Rocco A. Servedio, editors, *Proceedings of the 55th Annual ACM Symposium on Theory of Computing, STOC 2023, Orlando, FL, USA, June 20-23, 2023*, pages 595–608. ACM, 2023.

[66] Wei-Kai Lin, Ethan Mook, and Daniel Wichs. Black box crypto is useless for doubly efficient PIR. In Serge Fehr and Pierre-Alain Fouque, editors, *Advances in Cryptology - EUROCRYPT 2025 - 44th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Madrid,*

Spain, May 4-8, 2025, Proceedings, Part VI, volume 15606 of *Lecture Notes in Computer Science*, pages 65–93. Springer, 2025.

[67] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. On ideal lattices and learning with errors over rings. *J. ACM*, 60(6):43:1–43:35, 2013.

[68] Moni Naor. Bit commitment using pseudo-randomness. In Gilles Brassard, editor, *Advances in Cryptology – CRYPTO '89 Proceedings*, pages 128–136. Springer, 1990.

[69] Tal Rabin and Michael Ben-Or. Verifiable secret sharing and multiparty protocols with honest majority. In David S. Johnson, editor, *Proceedings of the Twenty-First Annual ACM Symposium on Theory of Computing (STOC 1989)*, pages 73–85. ACM, 1989.

[70] Adi Shamir. How to share a secret. *Communications of the ACM*, 22(11):612–613, 1979.

[71] David P. Woodruff and Sergey Yekhanin. A geometric approach to information-theoretic private information retrieval. In *20th Annual IEEE Conference on Computational Complexity (CCC 2005), 11-15 June 2005, San Jose, CA, USA*, pages 275–284. IEEE Computer Society, 2005.

[72] Sergey Yekhanin. Towards 3-query locally decodable codes of subexponential length. *J. ACM*, 55(1):1:1–1:16, 2008.

A Hadamard-based toy example

We begin by introducing a simplified two-server RC-PIR construction A.1. Although this construction incurs high communication cost, it establishes the structural framework for all subsequent constructions.

Construction A.1. *Let $n \in \mathbb{N}$ the size of the database, $m = m(n)$ the number of samples, $s = s(n)$ the sparsity setting. Let $H \xleftarrow{R} \{0,1\}^{n\times m}$ be a public random matrix.*

1. Database Encoding: *On input database $x \in \{0,1\}^n$, output $R = x^T H \in \{0,1\}^{1\times m}$.*
2. Query: *The query algorithm* $\text{QUERY}(i;\rho)$*, outputs a tuple (q_r, q_c) where $q_r \subseteq [m]$ is a uniformly random subset of requested entries such that $|q_r| = s$ and $q_c = H \cdot 1_{q_r} + 1_i$.*
3. Retrieval Answer: *Enforced to output $\mathbf{ans}_r^{(j)} = R_j$ for any given entry address $j \in q_r$.*
4. Computation Answer: *When the computation server receives query q_c, the algorithm* $\text{CANSWER}(x, q_c)$*, outputs $\mathbf{ans}_c = x^T q_c$.*
5. Decoding: *The decoding algorithm* $\text{DEC}(i, r, \mathbf{ans}_r, \mathbf{ans}_c)$ *where $\mathbf{ans}_r = \left\{\mathbf{ans}_r^{(j)}\right\}_{j\in q_r}$ and outputs $\mathbf{ans}_c - \sum_{j\in q_r} \mathbf{ans}_r^{(j)}$.*

Lemma A.2. *The toy example is perfectly correct.*

Proof. Let $x \in \{0,1\}^n$ be a database of size n, and let $H \in \{0,1\}^{n \times m}$ be an arbitrary matrix. Then, for every index $i \in [n]$ whose generated retrieval query is q_r, satisfy

$$\begin{aligned}
\mathbf{ans}_c - \sum_{j \in q_r} \mathbf{ans}_r^{(j)} &= x^T q_c - \sum_{j \in q_r} R_j \\
&= x^T \cdot (H \cdot 1_{q_r} + 1_i) - \sum_{j \in q_r} (x^T H)_j \\
&= x^T \cdot (H \cdot 1_{q_r} + 1_i) - x^T \cdot H \cdot 1_{q_r} \\
&= x^T \cdot 1_i \\
&= x_i.
\end{aligned}$$

Hence, the scheme is perfectly correct. □

Lemma A.3. *The toy example is secure under (n, m, s)-dLPN assumption.*

Proof. The privacy from the retrieval server perspective is trivial, as it is independent of the secret index. From the perspective of the computation server, the adversary's view contains a query of the form $H \cdot 1_{q_r} + 1_i$ where 1_{q_r} is a random s-sparse vector and $H \in \{0,1\}^{n \times m}$ is a public random matrix. Thus, under the (n, m, s)-dLPN assumption

$$(H, H \cdot 1_{q_r} + 1_i) \approx_c (H, r)$$

where $r \xleftarrow{R} \{0,1\}^n$ is uniformly sampled. □

In the toy example, although the construction satisfy the definition of an RC-PIR scheme, it results with substantial communication costs. The computation server receives a query of length n bits, while the retrieval server receives a query of length $s \log m$ bits. Their corresponding answers have lengths of 1 bit and s bits (respectively). Overall, this yields a total communication cost of $O(sn)$. We treat the public matrix H as a public global parameter that can be reused in various systems and so we do not count it as client storage.

Improving Communication Cost. To reduce the communication cost, we interpret the database $x \in \{0,1\}^n$ as a collection of $N = \sqrt{n}$ smaller databases of size N each i.e. $X = \{0,1\}^{N \times N}$. We then reuse the same matrix $H \in \{0,1\}^{N \times m}$ and the same queries q_r, q_c for all databases allowing each server to run its answer algorithm on all databases using identical parameters. As a result, the retrieval server responds with sN bits and the computation server responds with N bits while their corresponding queries have length $s \log m$ bits and N bits, respectively.

Construction A.4 (Optimized Toy Example)**.** *Let $n = N^2$ the database size and let $X \in \{0,1\}^{N \times N}$ represent the database. Let $m = m(N)$ the number of samples, $s = s(N)$ the sparsity setting. Let $H \xleftarrow{R} \{0,1\}^{N \times m}$ be a public pseudorandom matrix.*

1. Database Encoding: *On input database $X \in \{0,1\}^{N \times N}$, output $R = XH$.*
2. Query: *The query algorithm* $\textsc{Query}(i = (a, b); \rho)$*, outputs a tuple (q_r, q_c) where $q_r \subseteq [m]$ is a uniformly random subset of requested entries such that $|q_r| = s$ and $q_c = H \cdot 1_{q_r} + e_a$.*
3. Retrieval Answer: *Enforced to output $\mathbf{ans}_r^{(j)} = R_j \in \mathbb{F}^N$ for any given entry address $j \in q_r$.*

4. Computation Answer: *When the computation server receives query q_c, the algorithm* $\text{CANSWER}(C, q_c)$, *outputs* $\mathbf{ans}_c = X \cdot q_c$.

5. Decoding: *The decoding algorithm* $\text{DEC}(i = (a, b), r, \mathbf{ans}_r, \mathbf{ans}_c)$ *where* $\mathbf{ans}_r = \left\{\mathbf{ans}_r^{(j)}\right\}_{j \in q_r}$, *outputs* $\langle \mathbf{ans}_c - \sum_{j \in q_r} \mathbf{ans}_r^{(j)}, e_b \rangle$.

Lemma A.5. *The optimized toy example is perfectly correct.*

Proof. Let $X \in \{0,1\}^{N \times N}$ be a database of size $n = N^2$, and let $H \in \{0,1\}^{n \times m}$ be an arbitrary random matrix. Then, for every index $i = (a, b) \in N \times N$ whose generated retrieval query is q_r, satisfy

$$\begin{aligned}
\mathbf{ans}_c - \sum_{j \in q_r} \mathbf{ans}_r^{(j)} &= X \cdot q_c - \sum_{j \in q_r} R_j \\
&= X \cdot (H \cdot 1_{q_r} + e_a) - \sum_{j \in q_r} (X \cdot H)_j \\
&= X \cdot (H \cdot 1_{q_r} + e_a) - X \cdot H \cdot 1_{q_r} \\
&= X \cdot e_a \\
&= X_a
\end{aligned}$$

Thus,

$$\langle \mathbf{ans}_c - \sum_{j \in q_r} \mathbf{ans}_r^{(j)}, e_b \rangle = \langle X_a, e_b \rangle = X_{a,b}$$

Hence, the scheme is perfectly correct. □

The optimized toy example is secure under the (N, m, s)-dLPN assumption. The retrieval server query length is $s \log m$ and the computation query length is N. The answer length is sN for the retrieval server and N for the computation server. This result with a total communication cost of $O(sN) = O(s\sqrt{n})$.

B Computational Complexity of Sparse MV's

In this section, we prove Theorem 3.8 and show that the MV construction promised in Theorem 3.5 has computational complexity $O_d(h)$. That is, given an index $i \in [n]$, one can compute the vectors $u_i \in \mathbb{Z}_6^h$ and $v_i \in \mathbb{Z}_6^h$ using circuits of size $O_d(h)$.

Background. We begin by reviewing the necessary background on the construction of sparse matching vectors. In what follows, we set $m = 6$, with $p_1 = 2$ and $p_2 = 3$. Let $S_{\mathsf{can}} \subset \mathbb{Z}_6$ denote the set of integers whose CRT representation (with respect to 2 and 3) lies in the set $\{11, 01, 10\}$. Recall that for integers h, n, and d, a $(d, 2)$-sparse S_{can}-matching vector family consists of n pairs of vectors $u_i, v_i \in \mathbb{Z}_m^h$ for $i \in [n]$, where both u_i and v_i are d-sparse modulo 2, satisfying $\langle u_i, v_i \rangle = 0 \pmod m$, and for all $i \neq j$, $\langle u_i, v_j \rangle \in S_{\mathsf{can}}$. In [11], it is shown that for every $n, d > 0$, such a collection can be constructed with

$$h \leq d^{O(1)} n^{\frac{\log \log d}{\log d}}.$$

The construction uses several parameters (e.g., e_1, e_2, $t = 2^{e_1} \cdot 3^{e_2}$, and $\tilde{h} = \lceil n^{1/(t-1)} \rceil \cdot t$), which we leave partially unspecified here. A concrete instantiation of these parameters, as a function of n and d, can be found in [11, Lemma 4.15].

The matching vector family will be defined based on two main components. The first component is a set system $A_1, \ldots, A_n \subset [\tilde{h}]$ for which

$$|A_i \cap A_j| \equiv 0 \pmod{t} \quad \text{if and only if} \quad i = j. \tag{4}$$

The second component is a special "translation" polynomial $Q_t(x)$ over $\mathbb{Q}$ that translates $\mathbb{Z}_t$ to $\mathbb{Z}_m$ so that $0 \in \mathbb{Z}_t$ is mapped to $0 \in \mathbb{Z}_m$ and every other input in $\mathbb{Z}_t$ is mapped to S_{can}. The polynomial $Q_t(x)$ is taken from the work of [10]. The definition of u_i and v_i will be based on A_i and Q_t so that $\langle u_i, v_j \rangle = Q_t(|A_i \cap A_j|)$ via the reduction of [63].

The formal properties of the translation polynomial are summarized in the following theorem.

Theorem B.1 ([11] Theorem 4.10)**.** *Let $m = 6$ and $t = 2^{e_1}3^{e_2}$ for two positive integers e_1, e_2. There exists a univariate polynomial $Q_t(x)$ over $\mathbb{Q}$ of degree $d_Q = \max\{2^{e_1}, 3^{e_2}\} - 1$ such that:*

1. *$Q_t(x) = \sum_{i=1}^{d_Q} b_i \binom{x}{i}$, where $b_i \in \{0, \ldots, m-1\}$.*
2. *For every $x \in \mathbb{Z}_t$,*

$$Q_t(x) \pmod{m} \equiv 0 \quad \textit{if} \quad x \equiv 0 \pmod{t},$$

and

$$Q_t(x) \pmod{m} \in S_{\mathsf{can}} \quad \textit{if} \quad x \not\equiv 0 \pmod{t}.$$

Remark B.2 (The uniformity of Q_t)**.** *Jumping ahead, the vector $b = (b_1, \ldots, b_{d_Q})$ will be hardwired to the circuit that computes u_i and v_i. We note that this vector can be efficiently computed and therefore the circuit can be computed uniformly in polynomial time. Indeed, a closer look at Q_t (see, e.g., [63, Thm 3.1]) shows that for every i the scalar $b_i \in \{0, \ldots, 5\}$ is the unique scalar whose CRT representation satisfies*

$$b_i \equiv (-1)^{i+1} \pmod{2}, \quad \textit{and} \quad b_i \equiv (-1)^{i+1} \pmod{3}.$$

Therefore, we can compute the vector b in time polynomial (or even linear) in the length of b.

The following lemma combines the polynomial and the set system and defines an MV family (the proof is given for completeness).

Lemma B.3. *Let $A = \left\{A_j \subseteq [\tilde{h}]\right\}_{j \in [n]}$ be a collection of sets of size t that satisfies the intersection property as stated in (4), let $b = (b_1, \ldots, b_{d_Q})$ be the coefficients of the translation Polynomial (Theorem B.1).*

Let $h = \sum_{k=1}^{d_Q} \binom{\tilde{h}}{k}$ and for each $i \in [n]$ define the vectors $u_i, v_i \in \mathbb{Z}_6^h$ that are indexed by sets $\emptyset \neq S \subseteq [\tilde{h}]$ of size at most d_Q as

$$u_i[S] = b_{|S|} \cdot 1_{S \subseteq A_i}, \quad v_i[S] = 1_{S \subseteq A_i}.$$

Then, the collection (u_i, v_i) is an S_{can}-matching vector family of dimension h and sparsity $d = \sum_{k=1}^{d_Q} \binom{t}{k}$.

Proof. Since for every A_i, $|A_i| = t$ the number of non-zero coordinates in u_i, v_i is the number of non-empty subsets A_i of size at most d_Q, i.e. $d = \sum_{k=1}^{d_Q} \binom{t}{k}$. Similarly, the length of the vectors is $h = \sum_{k=1}^{d_Q} \binom{\tilde{h}}{k}$. The inner product is given by:

$$\langle u_i, v_j \rangle = \sum_{k=1}^{d_Q} \sum_{S \in \binom{[\tilde{h}]}{k}} u_i[S] \cdot v_j[S] = \sum_{k=1}^{d_Q} b_k \sum_{S \in \binom{[\tilde{h}]}{k}} 1_{S \in A_i \cap A_j} = \sum_{k=1}^{d_Q} b_k \binom{|A_i \cap A_j|}{k} = Q_t(|A_i \cap A_j|).$$

By the properties of Q_t and the intersection property of A, if follows that $\langle u_i, v_j \rangle \equiv 0 \pmod{m}$ if and only if $i = j$, and $\langle u_i, v_j \rangle \mod m \in S_{\mathsf{can}}$ otherwise. □

The lemma shows that the computational complexity of the MV family is dominated by the complexity of the set system. Formally, we have the following proposition.

Proposition B.4 (From sets to vectors)**.** *Let A and b be as in Lemma B.3. Let T be a circuit that given $i \in \{0,1\}^{\lceil \log n \rceil}$ outputs the characteristic vector $1_{A_i} \in \{0,1\}^{\tilde{h}}$. Then, the MV-selcttion circuits U and V can be computed by a circuit of size $|T| + O_d(h)$.*

Proof. The circuit V is defined as follows. Given i invoke $T(i)$ and generate 1_{A_i}. For every non-empty subset $S \subseteq [\tilde{h}]$ such that $|S| \leq d_Q$, we take the Sth output to be an AND-tree over the entries $1_{A_i}[j], j \in S$. The tree is of size at most $d_Q < d$. The circuit U is similar except that at the end we multiply the final output with the coefficients b that are hardwired to the circuit. Since the multiplication os over a fixed-size domain ($\mathbb{Z}_6$) the overhead is $O(h)$. □

It remains to explain how to define the set system and to analyze the complexity of selecting the ith set.

Constructing the sets. To achieve the *intersection property* $|A_i \cap A_j| \equiv 0 \pmod{t}$ if and only if $i = j$, we construct A as follows. Let $w = t - 1$ and $\tilde{h} = \left\lceil n^{1/w} \right\rceil \cdot w + 1$. Partition $[\tilde{h} - 1]$ into w blocks of size $B = \left\lceil n^{1/w} \right\rceil$ each. For every $i \in [n]$ we define the set A_i so that it contains at most a single element in each block B. Specifically, letting $(x_1, \ldots, x_w) \in [B]^w$ denote the representation of i in base B, i.e., $i = \sum_{k=1}^{w} x_k B^{k-1}$, we let

$$A_i := \{B \cdot (k-1) + x_k \mid k \in [w]\} \cup \{0\}\,,$$

where the "special" element 0 is being added to all the sets. This ensures that:

1. Each set has size exactly $w + 1 = t$. Thus, $|A_i \cap A_i| = t \equiv 0 \pmod{t}$, for every $i \in [n]$.
2. For $i \neq j$, the set can intersect in at most $w - 1$ "block elements", plus the special element, meaning $1 \leq |A_i \cap A_j| \leq w = t - 1$. That is $|A_i \cap A_j| \not\equiv 0 \pmod{t}$.

The following proposition is immediate.

Proposition B.5 (Selecting a set)**.** *Let A be the above collection of sets. Then the circuit T that given $i \in \{0,1\}^{\lceil \log n \rceil}$ outputs the characteristic vector $1_{A_i} \in \{0,1\}^{\tilde{h}}$, can be implemented with size $O(\tilde{h}) = O(h)$.*

Proof. Let us first assume that B is a power of 2. We view the binary representation of i as w blocks of $\log B$ bits each. Each block $j \in [w]$ is wired into a decoder that takes $\log B$ inputs and output B bits, defining the j-th block of 1_{A_i}. This layer requires w decoders of size $O(B)$ each, thus the size of the first layer is $O(wB) = O(\tilde{h}) = O(h)$. Assume without loss of generality that B is a power of 2.

Next, we argue that we can always round B up to the next power of 2 by setting $B = 2^{\lceil \log n/w \rceil}$ without affecting the asymptotic relations between the parameters. Indeed, this adjustment increases B by a factor of at most 2, which correspondingly increases $\tilde{h} = B \cdot w + 1$ by a factor of at most 2. Consequently, h also increases by a factor of at most 2^{d_Q}. Since, $d \geq (\sqrt{2t})^{\sqrt{t/2}}$ and $d_Q \leq \sqrt{2t}$ in Kutin's construction ([11], Claim 4.14, eq. 11), we have $\log 2^{d_Q} = d_Q \leq \sqrt{2t} \log \sqrt{2t} \leq \log (d^2)$, implying that $2^{d_Q} < d^2$. Thus, it remains that $h \leq d^{O(1)} n^{O(\frac{\log \log d}{\log d})}$, as required. □

Theorem 3.8 now follows from the combination of Propositions B.4 and B.5.

C Proof of Theorem 5.1

Given a fully private k-server RC-PIR scheme

$$\Pi_k = (\textsc{Encoder}_k, \textsc{Query}_k, \textsc{CAnswer}_k, \textsc{RAnswer}_k, \textsc{Dec}_k)$$

and a 2-server RC-PIR scheme

$$\Pi_2 = (\textsc{Encoder}_2, \textsc{Query}_2, \textsc{CAnswer}_2, \textsc{RAnswer}_2, \textsc{Dec}_2),$$

we construct a $(k+1)$-server RC-PIR scheme Π_{k+1} as follows.

1. Let n denote the size of the database x. Define

   $$\ell_k = \ell_k(n), \qquad B_k = B_k(n), \qquad \rho_k = \rho_k(n),$$

   and

   $$B_2 = B_2(B_k(n)), \qquad \ell_2 = \ell_2(B_k(n)), \qquad \rho_2 = \rho_2(B_k(n)).$$

2. The encoding algorithm first computes the $\ell_k \times B_k$ encoded database

   $$R' \leftarrow \textsc{Encoder}_k(x).$$

   Then, for each row $u \in [\ell_k]$, it computes the $\ell_2 \times B_2$ encoded database

   $$R_u \leftarrow \textsc{Encoder}_2(R'_u),$$

   where R'_u denotes the uth row of R'. The final encoded database R is the $\ell_2\ell_k \times B_2$ matrix obtained by stacking the matrices

   $$R_1, \ldots, R_{\ell_k}$$

   one on top of another.

3. The query algorithm $\textsc{Query}(i; r)$ proceeds as follows. First, it samples a Π_k query tuple

   $$(q_1, \ldots, q_{k-1}, q'_k) \leftarrow \textsc{Query}_k(i; r_k),$$

   where r_k is fresh randomness extracted from r, and where the retrieval query q'_k consists of the block addresses

   $$(\alpha_1, \ldots, \alpha_{\rho_k}).$$

   Next, for each $j \in [\rho_k]$, the query algorithm samples a Π_2 query tuple

   $$(q_{1,j}, q_{2,j}) \leftarrow \textsc{Query}_2(\alpha_j; r_{2,j}),$$

   where $r_{2,j}$ is fresh randomness extracted from r.

   Observe that the queries $q_{2,j}$ are the "fetch" queries directed to the retrieval server of Π_2.

   Let

   $$q_k = (q_{1,j})_{j \in [\rho_k]} \quad \text{and} \quad q'_{k+1} = (q_{2,j})_{j \in [\rho_k]}.$$

   The query algorithm outputs the $(k+1)$-tuple

   $$(q_1, \ldots, q_k, q'_{k+1}).$$

4. For every $s \in [k-1]$, the sth computation server responds to the query q_s with

$$\mathbf{ans}_s = \mathrm{CANSWER}_k(s, x, q_s).$$

5. The kth computation server receives the query

$$q_k = (q_{1,j})_{j \in [\rho_k]}$$

and responds with

$$\mathbf{ans}_k = (\mathbf{ans}_{k,j,u})_{j \in [\rho_k],\, u \in [\ell_k]},$$

where

$$\mathbf{ans}_{k,j,u} = \mathrm{CANSWER}_2(R'_u, q_{1,j}).$$

6. The retrieval server receives the query

$$q'_{k+1} = (q_{2,j})_{j \in [\rho_k]},$$

where each $q_{2,j}$ consists of ρ_2 addresses

$$(\beta_{j,1}, \ldots, \beta_{j,\rho_2}) \in [B_2]^{\rho_2}.$$

The server responds with the corresponding columns of R, namely

$$\mathbf{ans}_{k+1} = (R[\beta_{j,u}])_{j \in [\rho_k],\, u \in [\rho_2]},$$

where $R[\beta]$ denotes the βth column (i.e., block) of R.

7. The decoding algorithm parses the randomness as

$$r = (r_k, (r_{2,j})_{j \in [\rho_k]}),$$

and, given the answers $(\mathbf{ans}_i)_{i \in [k+1]}$, proceeds as follows.

First, it reconstructs the answers of the "virtual" kth server of Π_k to the query

$$q'_k = (\alpha_1, \ldots, \alpha_{\rho_k}).$$

Specifically, for every $j \in [\rho_k]$, it computes the column

$$R'[\alpha_j] = (\mathrm{DEC}_2(\alpha_j, r_{2,j}, \mathbf{ans}_{k,j,u}, \mathbf{ans}_{k+1,j,u}))_{u \in [\ell_k]}. \tag{5}$$

It then assembles the recovered columns into $\mathbf{ans}'_k$ and invokes the decoder of Π_k:

$$\mathrm{DEC}_k(i, r_k, \mathbf{ans}_1, \ldots, \mathbf{ans}_{k-1}, \mathbf{ans}'_k). \tag{6}$$

Correctness. Denote by ϵ_k (resp., ϵ_2) the correctness error of Π_k (resp., Π_2). By the correctness of Π_2 and a union-bound, Eq. (5) holds except with probability $\epsilon_2(B_k(n)) \cdot \ell_k(n)$. Similarly, by the correctness of Π_k, Eq. (6) holds except with probability $\epsilon_k(n)$, and so the total error is at most $\epsilon_2(B_k(n)) \cdot \ell_k(n) + \epsilon_k(n)$. Since ϵ_k and ϵ_2 are negligible (by assumption) and since $B_k(n)$ and $\ell_k(n)$ are polynomial in n (by assumption), the overall error is negligible.

Privacy. We denote the query algorithm of Π_{k+1} by QUERY. Fix an index $i, i' \in [n]$ and a k-subset $T \subset [k+1]$, we will show that $\text{QUERY}(i)[T]$ is computationally indistinguishable from $\text{QUERY}(i')[T]$. Here $\text{QUERY}(i)[T]$ is the random variable

$$(q_j : j \in T) \quad \text{where} \quad (q_1, \ldots, q_{k+1}) \leftarrow \text{QUERY}(i).$$

First consider the case where one of the first $k-1$ servers does not participate in T. Without loss of generality, assume that the missing server is the first server. Recall that the view of the last two servers in Π_{k+1} is computed based on the view of the last (retrieval) server in Π_k. Therefore, the T-view in Π_{k+1} can be efficiently computed based on the view of $T' = \{2, \ldots, k\}$ in Π_k, i.e., $\text{QUERY}(i)[T] = f(\text{QUERY}_k(i)[T'])$ for some efficiently computable randomized function f. Since $\text{QUERY}_k(i)[T']$ is indistinguishable from $\text{QUERY}_k(i')[T']$ (by the privacy of Π_k), we conclude that $\text{QUERY}(i)[T]$ is indistinguishable from $\text{QUERY}(i')[T]$.

Next, assume that the one of the last two servers does not participate in T. For concreteness, say that this is the last $(k+1)$th server. (The other case is similar.) For $0 \le j \le \rho_k$, define the hybrid distribution $H_j(i)$ identically to $\text{QUERY}(i)[T]$ except that the first j calls to QUERY_2 are replaced with calls to $\text{QUERY}_2(1)$. (That is, we generate Π_2 PIR queries that corresponds to a request for the first index.) By the privacy of Π_2 each neighboring hybrids $H_j(i)$ and $H_{j+1}(i)$ are computationally indistinguishable. By definition, $H_0(i) \equiv \text{QUERY}(i)[T]$, and therefore, it suffices to show that the final hybrid hides i, i.e., that $H_{\rho_k}(i)$ is indistinguishable from $H_{\rho_k}(i')$.

Indeed, in the final hybrid, $H_{\rho_k}(i)$, the queries generated for the kth server are independent of i and therefore this hybrid can be efficiently computed based on $\text{QUERY}_k(i)[T']$ where $T' = [k-1]$. Since $\text{QUERY}_k(i)[T']$ is indistinguishable from $\text{QUERY}_k(i')[T']$ (by the privacy of Π_k), we conclude that $H_{\rho_k}(i)$ is is indistinguishable from $H_{\rho_k}(i')$, as required. This completes the proof of Theorem 5.1. □

D Proof of Thm. 5.7

Let Π_k denote a k-party PIR scheme that realizes some Q_3 access structure Γ over the servers $P = \{p_1, \ldots, p_k\}$. We define the robust decoder as follows: Given an answer vector $(\mathbf{ans}_1, \ldots, \mathbf{ans}_k)$ and randomness r (that was used to generate the queries), the decoder output $b \in \{0,1\}$ if there exists a maximal unauthorized set $C \subseteq P$, such that for all the authorized subsets S that are disjoint from C, decoding over S, i.e., $\text{DEC}_S(i, r, (\mathbf{ans}_i)_{i \in S})$, leads to the value b. Equivalently, letting $H = P \setminus C$ denote the complement of C, we enumerate over all authorized sets $S \subseteq H$. If the above condition holds for both 0 and 1, we output a special failure symbol.

We now prove that the decoder is correct. Let us condition on the event that, for every authorized uncorrupted set, the original decoder does not err (which happens w/p at least $1 - \delta 2^t$). Without loss of generality, assume the true database bit is 0. Let C_0 denote the byzantine servers controlled by the adversary. By definition, C_0 is an unauthorized set. Because the true bit is 0, any authorized subset of the honest servers $H_0 = P \setminus C_0$ will reconstruct 0. Assume towards a contradiction that there exists an unauthorized set C_1 such that all authorized subsets of $H_1 = P \setminus C_1$ reconstruct to 1 and let $C_{\mathsf{both}} = H_0 \cap H_1$. To derive a contradiction, we show that C_{both} cannot be neither authorized nor unauthorized. Indeed, if C_{both} was an authorized set, its answers would decode to 0 (since $C_{\mathsf{both}} \subseteq H_0$) and simultaneously to 1 (since $C_{\mathsf{both}} \subseteq H_1$), which is impossible. On the other hand, if C_{both} is unauthorized, then by De Morgan's laws $C_{\mathsf{both}} = (P \setminus C_0) \cap (P \setminus C_1) = P \setminus (C_0 \cup C_1)$, and we can cover P by the union of three unauthorized sets:

$$C_0 \cup C_1 \cup C_{\mathsf{both}} = P,$$

in contradiction to the assumption that Γ is a Q_3 access structure.

Finally, we implement the robust decoder in time $O(2^k T + 2^{2k})$ as follows. First in time $O(2^k T)$ apply standard decoding with respect to every authorized subset S, and keep the resulting values b_S. Next, for each maximal unauthorized set C, we check whether b_S agree over all subsets $S \subseteq P \setminus C$ in time 2^{2k}, as required.

E MPC Proofs

E.1 Proof of Lemma 6.2

Correctness follows immediately from the correctness of the PIR. We proceed with the privacy proof.

Consider a passive adversary controlling a strict subset T of the parties. (The case where $T = [k]$ is trivial.) The simulator receives inputs $y_T = \{y_j\}_{j \in T}$ and the final output $z = f(y)$ and proceeds as follows:

1. Sample local randomizers $(\hat{r}_i)_{i \in [k]}$, let $\hat{r} = \sum_i \hat{r}_i$ and choose an arbitrary vector $\hat{y}$ satisfying $y_T = \hat{y}_T$.
2. Compute $(\hat{q}_1, \ldots, \hat{q}_k) \leftarrow \textsc{Query}(\hat{y}, \hat{r})$ and output the view of the adversary

$$y_T, \quad \hat{r}_T = (\hat{r}_j)_{j \in T}, \quad \hat{q}_T = (\hat{q}_j)_{j \in T}, \quad \text{and} \quad z.$$

   We omit from the notation the locally computed answers $\mathbf{ans}_j = \textsc{Ans}(F, q_j)$ for $j \in T$, since these are deterministic functions of the public table F and the queries q_T. Including them in the adversary's view would not affect the argument.

We show that the simulated view is computationally indistinguishable from the real view in the hybrid model. Specifically, for every truth-table F and for every target index y, and every coalition $T \subset [k]$, let $z = F[y]$, the following views are indistinguishable,

$$(y_T, r_T, q_T, z) \approx_c (y_T, \hat{r}_T, \hat{q}_T, z).$$

Indeed, we can write

$$(y_T, r_T, q_T, z) = A(q_T) \quad \text{and} \quad (y_T, \hat{r}_T, \hat{q}_T, z) = A(\hat{q}_T)$$

where $A = A_{T, y_T, z}$ is the efficient randomized algorithm that, on input q_0, samples r_T uniformly and outputs (y_T, r_T, q_0, z). Since A is efficient and since q_T and $\hat{q}_T$ are computationally indistinguishable (by the privacy of the PIR), we conclude that the simulated view and real view are indistinguishable as well. The lemma follows. □

E.2 Proof of Lemma 6.5

Fix a corrupted coalition $T \subseteq [k]$ of size $t' \leq t$, where $k = 2t+1$, and let $\bar{T} = [k] \setminus T$. Given a $\mathrm{poly}(n)$-size adversary $\mathcal{A}$, the simulator invokes $\mathcal{A}$ and proceeds as follows.

1. The adversary $\mathcal{A}$ sends its g_1-inputs (y_T, r_T).

2. The simulator sets $\hat{y}_{\bar{T}} = \mathbf{0}$, samples $r_{\bar{T}}$ uniformly at random, and honestly emulates g_1 on inputs $\hat{y} = (y_T, \hat{y}_{\bar{T}})$ and randomness $(r_T, r_{\bar{T}})$. Specifically, it sets $r = \sum_i r_i$, computes $(q_S)_{S \in \binom{[k]}{t}} = \text{QUERY}(\hat{y}; r)$, and generates the shares

$$(Q_{j,S})_{j \in [k]} = \text{SHARE}(q_S) \quad \forall S \in \binom{[k]}{t}, \qquad (C_{j,i})_{i \in [k]} = \text{SHARE}(y_i, r_i) \quad \forall i \in [k].$$

For every $j \in T$, the simulator sends to the adversary

$$(Q_{j,S})_{S \in \binom{[k]}{t}}, \qquad (C_{j,i})_{i \in [k]}, \qquad (q_S)_{S:\, j \notin S}.$$

3. The adversary sends

$$(\mathbf{ans}_{j,S})_{S:\, j \notin S}, \qquad (C'_{j,i})_{i \in [k]} \qquad \forall j \in T.$$

For every $S \in \binom{[k]}{t}$ and every honest party $i \in \bar{T} \setminus S$, the simulator sets $\mathbf{ans}_{i,S} := \text{ANS}(F, q_S)$.

If, for every S, all values $(\mathbf{ans}_{j,S})_{j \notin S}$ agree, the simulator delivers y_T to the ideal functionality, receives the output $\hat{z}$, sends $\hat{z}$ to the adversary, and terminates.

Otherwise, the simulator sends to the adversary the lexicographically first conflict tuple

$$(i, i', S, \mathbf{ans}_{i,S}, \mathbf{ans}_{i',S})$$

such that $\mathbf{ans}_{i,S} \neq \mathbf{ans}_{i',S}$.

4. For every $j \in \bar{T}$, the simulator sends the share $Q_{j,S}$ to the adversary. The adversary responds with shares $Q'_{j,S}$ for every $j \in T$. The simulator applies robust recovery to the shares $(Q'_{j,S})_{j \in T}$ and $(Q_{j,S})_{j \in \bar{T}}$, reconstructing q'_S. It then computes $\mathbf{ans}'_S = \text{ANS}(F, q'_S)$. If $\mathbf{ans}_{i,S} \neq \mathbf{ans}'_S$, the simulator sends the identifiable abort $(\perp, \{i\})$ to the ideal functionality. Otherwise, it sends $(\perp, \{i'\})$. (Recall that $\mathbf{ans}_{i,S} \neq \mathbf{ans}_{i',S}$; therefore, in the latter case, $\mathbf{ans}_{i',S} \neq \mathbf{ans}'_S$.)

The simulator is well defined. We claim that the simulator is well defined, in the sense that every identifiable abort sent to the ideal functionality names a corrupted party in T. Indeed, in the last step of the simulation, the honest parties provide correct shares of q_S, whereas the adversary controls at most t shares. By the robustness of the secret-sharing scheme, the reconstructed value q'_S is therefore equal to the original query q_S, regardless of the adversary's strategy. Hence, for every honest party $i \in \bar{T}$,

$$\mathbf{ans}_{i,S} = \text{ANS}(F, q_S) = \text{ANS}(F, q'_S) = \mathbf{ans}'_S.$$

Therefore, the condition $\mathbf{ans}_{i,S} \neq \mathbf{ans}'_S$ can never hold for an honest party i. It follows that the simulator never forwards the identity of an honest party to the ideal functionality, and so every identifiable abort points to a corrupted party.

The real execution is indistinguishable from the ideal execution. Fix the inputs $y_{\bar{T}}$ of the honest parties. The analysis proceeds via a standard hybrid argument. Let H_0 denote the simulated view of the adversary concatenated with the output of the honest parties. Let H_1 be the hybrid distribution obtained by modifying the simulator so that it uses the real values $y_{\bar{T}}$ in its call to g_1, rather than the dummy values $\hat{y}_{\bar{T}} = \mathbf{0}$.

Claim E.1. *The hybrids H_0 and H_1 are computationally indistinguishable.*

Proof of claim. Let $T^\star \in \binom{[k]}{t}$ be an arbitrary set such that $T \subseteq T^\star$. The adversary never receives the query $q_{T^\star}$ in the clear, and receives only the shares $(Q_{j,T^\star})_{j\in T}$, which are perfectly hiding by the privacy of the secret-sharing scheme. We also note that the conflict-resolution phase never reveals a query q_S with $T \subseteq S$. Indeed, if $T \subseteq S$, then every party in $[k]\setminus S$ is honest, and hence all values $(\mathbf{ans}_{j,S})_{j\notin S}$ are honestly computed and agree. Thus, a conflict can occur only for sets S such that $T \nsubseteq S$. For every such S, there exists a corrupted party $j \in T \setminus S$, and so the adversary already received q_S in the clear during the call to g_1. Consequently, the adversary's entire view, including the values revealed during conflict resolution, can be generated from all PIR queries except $q_{T^\star}$, together with perfectly hiding shares of $q_{T^\star}$. By the full privacy of the PIR scheme and the perfect privacy of the secret-sharing scheme, H_0 and H_1 are computationally indistinguishable. □

It remains to compare H_1 with the real execution. By construction, the two distributions are identical up to Step 3. Fix the common transcript up to this point, including the adversary's input to g_2 and its internal random coins.

Observe that the answers of the honest parties are identical in H_1 and in the real execution. Moreover, for every $S \in \binom{[k]}{t}$, there exists at least one honest party in $\bar{T} \setminus S$, since

$$|\bar{T} \setminus S| \geq k - |T| - |S| \geq k - 2t > 0.$$

Thus, if no conflict occurs, the common value $\mathbf{ans}_S$ must equal $\textsc{Ans}(F, q_S)$ for every S, and the outputs in the two experiments can differ only if the event

$$E: \quad f(y) \neq \textsc{Dec}(y', r', \mathbf{ans})$$

occurs, where

$$(y'_i, r'_i) = \textsc{Rec}((C_{j,i})_{j\in[k]}) \qquad \forall i \in [k],$$

and $y' = (y'_i)_{i\in[k]}$, $r' = \sum_i r'_i$, and $\mathbf{ans} = (\mathbf{ans}_S)_{S\in\binom{[k]}{t}}$.

Since the honest parties provide at least $k - t' \geq k - t > t$ correct shares, the robustness of the secret-sharing scheme guarantees that $y' = y$ and $r' = r$, regardless of the adversary's strategy. Therefore,

$$\textsc{Dec}(y', r', \mathbf{ans}) = \textsc{Dec}(y, r, \mathbf{ans}),$$

and by the correctness of the PIR scheme, the event E occurs only with negligible probability over the choice of r (which is uniform since $(r_i)_{i\in\bar{T}}$ is uniform).

If a conflict occurs, then the conflict tuple is identical in both experiments, since the transcript up to Step 3 has been fixed. Moreover, the remainder of the execution is a deterministic function of values that have already been fixed. Hence, the outputs in the two experiments are identical.

It follows that H_1 and the real execution differ only with negligible probability. Combined with the indistinguishability of H_0 and H_1, this completes the proof. □

E.3 Proof of Lemmas 6.5 and 6.6

The proofs of Lemma 6.5 and Lemma 6.6 are presented simultaneously, as they follow essentially the same line of argument. Whenever the proofs differ, we explicitly point out and explain the necessary modifications.

The simulator. Fix a coalition $T \subset [k]$ of size $t' < k$ and let $\bar{T} = [k] \setminus T$. (The case where $T = [k]$ is trivial.) Given a $\mathrm{poly}(n)$-size adversary $\mathcal{A}$, the simulator $\textsc{Sim}_3$ (for $\Pi_3^{g_1,g_2}$) invokes $\mathcal{A}$ and proceeds as follows.

1. The adversary $\mathcal{A}$ sends its g_1-inputs (y_T, r_T).

2. The simulator sets $\hat{y}_{\bar{T}} = \mathbf{0}$, samples $r_{\bar{T}}$ uniformly at random, and honestly emulates g_1 on inputs $\hat{y} = (y_T, \hat{y}_{\bar{T}})$ and randomness $r = (r_T, r_{\bar{T}})$. Specifically, it computes

$$(q_1, \ldots, q_k) = \textsc{Query}(\hat{y}; \sum_i r_i)$$

   and

$$Q_i = \textsc{Com}(\hat{y}_i, r_i, q_i; o_i), \qquad \forall i \in [k],$$

   where the openings o_i are chosen uniformly at random. The simulator delivers (q_T, o_T) to the adversary together with the commitment vector $Q = (Q_i)_{i \in [k]}$.

3. The adversary sends to g_2 a vector $(y'_T, r'_T, \mathbf{ans}'_T)$. The simulator sets

$$y'_{\bar{T}} = \mathbf{0}, \qquad \mathbf{ans}'_{\bar{T}} = \mathbf{0}, \qquad r'_{\bar{T}} = r_{\bar{T}},$$

   and computes commitments

$$A_i = \textsc{Com}(y'_i, r'_i, \mathbf{ans}'_i; a_i), \qquad \forall i \in [k],$$

   using fresh random openings a_i.

   In addition, the simulator samples a signing/verification key pair $(\mathsf{sk}, \mathsf{vk})$ and sends

$$\mathsf{vk}, \qquad z_i, \qquad \zeta_i = \textsc{Sign}_{\mathsf{sk}}(i, z_i), \qquad \forall i \in T,$$

   where the shares $(z_i)_{i \in T}$ are sampled as a random sharing of 0 (in the additive secret-sharing case, this simply means choosing them independently and uniformly at random). The simulator sends the public values vk and the vector of commitments $A = (A_i)_{i \in [k]}$ to the adversary, and for every $i \in T$ delivers

$$(z_i, \zeta_i, a_i), \qquad \text{where} \quad \zeta_i = \textsc{Sign}_{\mathsf{sk}}(i, z_i).$$

4. For every pair $i \neq j$:

   - If $i \in \bar{T}$ and $j \in T$, the simulator proves Eq. (3) using the zero-knowledge simulator.
   - If $i \in T$ and $j \in \bar{T}$, the simulator verifies the adversary's proof and sets $b_{i,j} = 1$ if verification succeeds.
   - If $i, j \in T$, the adversary generates the proof transcript and verification bit $b_{i,j}$.
   - If $i, j \in \bar{T}$, simulator generates the proof transcript using the zero-knowledge simulator and sets $b_{i,j} = 1$.

   For notational convenience, set $b_{i,i} = 1$ for every $i \in [k]$.

5. For every $i \in [k]$ let $b_i = \wedge_j b_{i,j}$ and let $b = \wedge_i b_i$.

- If $b = 0$, the simulator instructs the ideal functionality to abort the honest parties.
- Otherwise, the simulator sends y_T to the ideal functionality and receives the output $\hat{z}$. It then samples shares $z_{\bar{T}}$ conditioned on

$$(z_T, z_{\bar{T}}) = \text{SHARE}(\hat{z}),$$

which can be done efficiently for any linear secret-sharing scheme. Next, it signs the honest shares

$$\zeta_i = \text{SIGN}_{\mathsf{sk}}(i, z_i), \qquad \forall i \in \bar{T},$$

and sends $(z_{\bar{T}}, \zeta_{\bar{T}})$ to the adversary.

The adversary responds with shares z'_T and signatures ζ'_T. If all signatures verify with respect to vk, the simulator instructs the ideal functionality to deliver the output $\hat{z}$ to the honest parties. Otherwise, it instructs the ideal functionality to abort the honest parties.

The simulator SIM_4 (for $\Pi_4^{g_1,g_2}$) is defined identically except that the adversary corrupts only $t' < t$ parties where $k = 2t + 1$, and the last step is defined as follows.

5. For $i \in [k]$, let $b_i = \text{MAJ}(b_{i,1}, \ldots, b_{i,k})$ and let $b = \wedge_i b_i$.
   - If $b = 0$, the simulator sets $B = \{i : b_i = 0\}$, sends $(\perp, B)$ to the ideal functionality, and terminates.
   - Else, (if $b = 1$) the simulator sends y_T to the ideal functionality and receives back the output $\hat{z}$. It then sample shares $z_{\bar{T}}$ conditioned on $(z_T, z_{\bar{T}}) \xleftarrow{R} \text{SHARE}(\hat{z})$, signs these shares $\zeta_i = \text{SIGN}_{\mathsf{sk}}(i, z_i), \forall i \in \bar{T}$, sends $(z_{\bar{T}}, \zeta_{\bar{T}})$ to the adversary, who answers with (z'_T, ζ'_T). The simulator terminates.

Observe that SIM_4 never points to an honest party $P_i, i \in \bar{T}$. Indeed, P_i's proofs are successfully accepted by every other honest party $P_j, j \in \bar{T}$, i.e., $b_{i,j} = 1$. (Here we assume, wlog, that the simulator always generates accepting transcripts). Since we have honest majority, we conclude that $b_i = \text{MAJ}(b_{i,1}, \ldots, b_{i,k}) = 1$ and therefore $i \notin B$, as claimed.

Analysis. We proceed by analyzing the simulators SIM_3 and SIM_4. Fix the inputs $y_{\bar{T}}$ of the honest parties. The analysis proceeds via a standard hybrid argument.[18]

Let H_0 denote the simulated view of the adversary concatenated with the output of the honest parties. Let H_1 be the hybrid distribution obtained by modifying the simulator so that it uses the real values $y_{\bar{T}}$ in its call to g_1, rather than the dummy values $\hat{y}_{\bar{T}} = \mathbf{0}$. By the privacy of the PIR scheme and the hiding property of the commitments Q_i, the distributions H_0 and H_1 are computationally indistinguishable.

Next, let H_2 denote the hybrid H_1 except that, in Step 3, the simulator sets $y'_{\bar{T}} = y_{\bar{T}}$ and $\mathbf{ans}'_{\bar{T}} = (\text{ANS}(F, q_j))_{j \in \bar{T}}$. This hybrid is computationally indistinguishable from H_1 by the hiding property of the commitments A_i.

Next, let H_3 denote the hybrid H_2 except that the zero-knowledge proofs generated by the simulator are now generated by the honest prover strategy. (Note that the simulator knows the corresponding witnesses.) A standard hybrid argument together with the zero-knowledge property implies that H_2 and H_3 are computationally indistinguishable.

[18] The hybrids are defined by gradually modifying the simulator. Unless stated otherwise, the same modifications are applied to SIM_3 and SIM_4 implicitly yielding two sequences of hybrids, one for SIM_3 and one for SIM_4.

Next, let H_4 be identical to H_3 except that $\hat{z}$ is computed as in the real protocol, namely

$$\hat{z} = \textsc{Dec}(y, r, \mathbf{ans}').$$

Let E denote the event that $b = 1$. First observe that conditioned on $\neg E$, the output of H_4 is distributed identically to the output of H_3. Indeed, in this case, in both hybrids, neither the adversary's view nor the output of the honest parties depends on $\hat{z}$. We therefore focus on the event E.

Claim E.2. *The distributions $[H_3 \mid E]$ and $[H_4 \mid E]$ are computationally indistinguishable.*

Proof of Claim. Let E_1 denote the event that E occurs and there exists a corrupted party P_i whose statement is false and nevertheless $b_i = 1$. By computational soundness of the argument system, the event E_1 occurs with negligible probability. Indeed, in Π_3, $b_i = 1$ implies that every verifier accepts P_i's proof. In Π_4, $b_i = 1$ implies that a majority of verifiers accept P_i's proof; since the adversary corrupts only a minority of the parties, at least one honest verifier must accept. Thus, in either case, E_1 implies that some honest verifier accepts a false proof, which occurs with negligible probability by soundness.

Since E_1 happens with negligible probability, it suffices to analyze the execution conditioned on $E_2 = E \setminus E_1$. Under this conditioning, every statement proved by a corrupted party and accepted by the protocol is true. Hence, for every $i \in T$, there exist witnesses $(\tilde{y}_i, \tilde{r}_i, \tilde{q}_i, \tilde{\mathbf{ans}}_i)$ and openings $(\tilde{o}_i, \tilde{a}_i)$ satisfying Eq. (3) with respect to the commitments Q_i and A_i. Since the commitments Q_i and A_i were honestly generated by the simulator, the statistical binding property implies that, except with negligible probability over the coins used to generate these commitments,

$$\tilde{y}_T = y_T = y'_T, \qquad \tilde{r}_T = r_T = r'_T, \qquad \tilde{q}_T = q_T, \qquad \tilde{\mathbf{ans}}_T = \mathbf{ans}'_T.$$

Moreover,

$$\mathbf{ans}'_i = \tilde{\mathbf{ans}}_i = \textsc{Ans}(F, \tilde{q}_i) = \textsc{Ans}(F, q_i), \qquad \forall i \in T.$$

Conditioned on the above, the value $\hat{z}$ computed in H_4 satisfies

$$\hat{z} = \textsc{Dec}(y, r, \mathbf{ans}'),$$

where

$$\mathbf{ans}'_i = \textsc{Ans}(F, q_i) \qquad \forall i \in [k],$$

and

$$(q_i)_{i \in [k]} = \textsc{Query}(y; r).$$

Since $r_{\bar{T}}$ is sampled uniformly at random by the simulator, the value $r = \sum_i r_i$ is uniformly distributed regardless of the adversary's choice of r_T. By the correctness of the PIR scheme, it follows that, except with negligible probability over the choice of r,

$$\hat{z} = f(y).$$

Condition on all the above events and fix all randomness in both experiments (including the adversary's randomness) except for the final step, in which $z_{\bar{T}}$ and $\zeta_{\bar{T}}$ are generated. Observe that in both hybrids these values are identically distributed conditioned on the fixed values z_T and the secret $z = \hat{z}$. Indeed, in both hybrids the values $z_{\bar{T}}$ are sampled according to the same conditional distribution defined by

$$(z_T, z_{\bar{T}}) = \textsc{Share}(\hat{z}).$$

This completes the proof of the claim. □

Finally, let H_5 be identical to H_4 except that, whenever the protocol does not abort, the output of the honest parties is defined as in the real execution of $\Pi_3^{g_1,g_2}$ and $\Pi_4^{g_1,g_2}$, respectively. Writing both protocols in a unified way, this means that

$$z = \text{REC}(S, (z_i)_{i \in S}), \quad \text{where} \quad S = (j : \text{VER}_{\mathsf{vk}}((j, z_j), \zeta_j) = 1).$$

(Note that in Π_3 the set S contains all the parties, and that $\text{REC}(S, (z_i)_{i \in S})$ is simply taken to be $\text{REC}((s_i)_{i \in [k]})$.)

Claim E.3. *The hybrids H_4 and H_5 are statistically-close.*

Proof of claim. By definition, the adversary's view in H_5 is distributed identically to its view in H_4. It therefore suffices to compare the outputs of the honest parties.

Fix the adversary's view and assume that all verifications succeed. We claim that, unless the adversary successfully forges a signature in the final step of the protocol (which occurs with negligible probability), the output z in H_5 equals the output $\hat{z}$ in H_4. Indeed, in this case, the adversary can only reveal the shares that were previously given to it by the simulator, and therefore $z'_i = z_i$ for every $i \in T \cap S$ where $S = (j : \text{VER}_{\mathsf{vk}}((j, z_j), \zeta_j) = 1)$ is of size at least k in $\Pi_3^{g_1,g_2}$ and larger than $k/2$ in $\Pi_4^{g_1,g_2}$. Since

$$(z_T, z_{\bar{T}}) = \text{SHARE}(\hat{z}),$$

the correctness of the secret-sharing scheme implies that

$$z = \text{REC}(S, z'_{T \cap S}, z_{\bar{T} \cap S}) = \text{REC}(S, z_S) = \hat{z}.$$

Therefore, conditioned on the adversary's view, the outputs in H_4 and H_5 differ only with negligible probability.[19] □

Finally, observe that H_5 is distributed identically to the real execution. This completes the proof of the lemma. □

[19] Recall that throughout this section the adversary is fixed to be polynomial-size. Thus, although the negligible bound follows from the computational unforgeability of the signature scheme, for the fixed adversary under consideration the two experiments differ only with negligible probability. Hence the induced distributions are statistically close.