Information-Theoretic Single-Server PIR in the Shuffle Model*

Yuval Ishai* Mahimna Kelkar† Daniel Lee§ Yiping Ma‡

Abstract

We revisit the problem of private information retrieval (PIR) in the *shuffle model*, where queries can be made anonymously by multiple clients. We present the first single-server PIR protocol in this model that has sublinear per-client communication and *information-theoretic security*. Moreover, following one-time preprocessing on the server side, our protocol only requires sublinear per-client *computation*. Concretely, for every $\gamma > 0$, the protocol has $O(n^\gamma)$ communication and computation costs per (stateless) client, with $1/\mathsf{poly}(n)$ statistical security, assuming that a size-n database is simultaneously accessed by $\mathsf{poly}(n)$ clients. This should be contrasted with the recent breakthrough result of Lin, Mook, and Wichs (STOC 2023) on doubly efficient PIR in the standard model, which is (inherently) limited to computational security.

1 Introduction

A private information retrieval (PIR) protocol [CGKS95, KO97] allows a client to fetch an entry from a database server without revealing which entry was fetched. Specifically, the server holds a database $x = (x_1, \ldots, x_n)$ consisting of n bits (or generically, n symbols over an alphabet Σ) while the client holds an index $i \in \{1, \ldots, n\}$; the client wishes to obtain x_i while hiding i from the server.

PIR protocols have been broadly studied in two flavors: information-theoretic and computational. Information-theoretic protocols provide security against computationally unbounded adversaries and do not require "cryptographic" computations. Unfortunately, non-trivial information-theoretic PIR (with less than n bits of communication) is impossible given only one server [CGKS95]. Consequently, PIR protocols in this setting need database replication across two or more non-colluding servers. This poses challenges for deployment since the cost of managing multiple storage spots is high when databases are large (e.g., synchronization, monetary cost), and enforcing non-collusion on the database servers is hard in practice, especially when the data is owned by a single entity (e.g., a company). In contrast, computational PIR can work when only one server holds the database but only provides security against polynomial-time adversaries due to its reliance on cryptographic hardness assumptions (e.g., quadratic residuosity, learning with errors). Furthermore, the associated cost is typically high due to expensive cryptographic operations at the server—indeed, existing single-server protocols [AMBFK16, ACLS18, ALP+21, MW22, HHCG+23, DPC23] are significantly less efficient in practice than the multi-server information-theoretic ones [GCM+16, GHPS22].

*Technion. yuvali@cs.technion.ac.il

†Cornell University and Cornell Tech. mahimna@cs.cornell.edu

‡MIT. lee_d@mit.edu

§University of Pennsylvania. yipingma@seas.upenn.edu

*This is a full version of [IKLM24].

The shuffle model: PIR with many clients. Achieving the best of both worlds, as aforementioned, is not possible in the standard model without using n bits of communication. To circumvent this barrier, Ishai, Kushilevitz, Ostrovsky and Sahai [IKOS06] proposed a relaxed model, where *many* clients (with arbitrarily correlated indices) simultaneously query a single server, but the clients are granted the ability to make anonymous queries to the server. Abstractly, we can think of the queries as being *shuffled* before reaching the server.

Specifically, consider a client using a *multi-server* PIR query algorithm to generate sub-queries for a query index. If these sub-queries were naively sent to a *single server*, the server would immediately learn the query index of this client. However, this work and [IKOS06] show the power of shuffling: if there are many clients and their sub-queries are randomly permuted by a shuffler before being sent to the server, then it is hard for the server—even one that is computationally unbounded as we show in this work—to figure out any of the client-query indices. Therefore, this single server in the shuffle model can simply perform "cheap" operations of the multi-server PIR scheme to answer sub-queries.

Understanding *the shuffle model* in the context of PIR is well-motivated by real-world applications: databases with high-volume queries, such as stock quotes and search engines, naturally enjoy the feature that thousands of users access the databases at the same time, and therefore considering PIR with many simultaneously querying clients is sensible, particularly if it allows for cheaper server cost. Note that this is a substantially different goal from batch PIR [IKOS04, Hen16] which amortizes the cost of multiple queries from a *single client* (see Section 3). The shuffle model has been considered also in problems orthogonal to PIR, including secure aggregation [IKOS06, BBGN20, GMPV20] and differential privacy [BEM+17, CSU+19, EFM+20, CU21, AIVG22]. Analogously to these works, we view shuffling as an *atomic* operation; existing literature on differential privacy [BBGN20] and anonymity [vdHLZZ15, LYK+19, APY20, DMS04, HSSN+22] discusses how to implement shuffling efficiently (see details in Section 1.2).

The shuffle PIR model opens a promising direction toward constructing efficient single-server PIR protocols. In this work, we establish the *theoretical feasibility* of non-trivial single-server PIR with information-theoretic security in the shuffle model.

1.1 Our Results

This paper aims to develop a formal understanding of PIR in the shuffle model from a theoretical perspective. We briefly detail our results below.

Information-theoretic single-server PIR in the shuffle model (Sections 5 and 6). We present the first construction for single-server PIR in the shuffle model that has *sublinear communication* and *information-theoretic* security (with inverse-polynomial statistical error). Moreover, our construction is also *doubly efficient*: following one-time preprocessing on the server side, and without any state information on the client side, the server's per-query computation is sublinear in the database size.

Theorem 1.1 (Informal). *For every constant $0 < \gamma < 1$, there exists a single-server PIR protocol in the shuffle model such that, on database of size n, and following one-time preprocessing on the server side, the protocol has $O(n^\gamma)$ per-query computation and communication, and $O(n^{1+\gamma/2})$ server storage. This is achieved with the following information-theoretic security guarantee: for any inverse polynomial $\epsilon = 1/p_1(n)$, there exists a polynomial $p_2(n) = O(n^{1+4/\gamma} \cdot (p_1(n))^8)$ such that* *the protocol has ϵ-statistical security as long as the total number of queries made by (uncorrupted) clients is at least $p_2(n)$.*

As a key technique, we describe a generic *inner-outer* paradigm that composes together two standard (multi-server) PIR protocols: an outer and an inner layer, to build a PIR protocol in the shuffle model. Besides, our results are robust against imperfect shuffling/anonymity (Appendix B).

While the above protocol only achieves inverse-polynomial (rather than negligible) security error, this is in fact the standard notion of security in several important settings, including differential privacy [DN03, DMNS06], secure computation with partial fairness [Cle86, MNS09, GK10], and secure computation over one-way noisy communication [AIK+21]. Our protocol demonstrates that *information-theoretic* security is indeed feasible *without database replication*. Moreover, while concrete efficiency is not the focus of this work, we believe that our approach has potential for reducing the cost of standard-model PIR when properly combined with single-server schemes and settling for a constant-factor cost reduction that might be significant in practice. See Section 6.4 for discussion.

Lower bound on security (Section 6.5). In the inner-outer paradigm, we show a security lower bound when *any* generic PIR protocol is used as the outer layer, and a *constant*-server PIR protocol from a broad class is used as the inner layer; in particular, $1/\mathsf{poly}(n)$ statistical security is tight in the sense that negligible security error cannot be achieved with polynomially many clients. We also discuss open problems (Section 7) on whether negligible security is possible (with polynomially many clients) by using other protocols in the inner layer.

1.2 Discussion on the Shuffle Model

Two-way shuffling. In the problems such as secure aggregation and differential privacy with shuffle model, the shuffled messages are delivered to a server for analytics. Our PIR setting is a bit different, since responses need to be communicated back from the server to the client, we require the shuffling to be *two-way*. Specifically, we require not only that clients can send messages anonymously to the server but also that the server can respond to clients while still keeping the client identities hidden. It is important to note that shuffling or anonymity does not trivialize the problem; it hides who sends the message but not the content itself. In practice, this two-way shuffling can be realized in a number of ways [KEB98, Cha81, DMS04, BBGN20, BBG23], even without computational assumptions.

A hybrid model. PIR in the shuffle model can also be equivalently viewed as a hybrid model between the standard single-server and multi-server PIR models: as an abstraction, the shuffler models a second “server” which is assumed to not collude with the main database server but does not hold a copy of the database and can only perform *database-irrelevant* computations. This alone makes the shuffle model interesting for practical deployments: non-collusion between two (or more) servers holding the same database can be difficult to enforce (since it is likely for them to be operated by the same company for data ownership reasons) making it a strong assumption in practice; in contrast, if only one server holds the database, then the “two” servers can be reasonably run by independent (and possibly geographically distributed) entities. We also note that it could be interesting to let this second database-irrelevant server perform more generic computations instead of just acting as a shuffler; we leave this exploration to future work.

2 Technical Overview

In this section, we present a toy protocol, which is insecure but conveys our core ideas; we then outline the techniques for building our eventual protocol from the toy protocol.

An insecure toy protocol. The starting point is the classic two-server information-theoretic PIR scheme by Beimel et al. [BIK05]. In this scheme, a client first deterministically encodes its queried index $i \in [n]$ to a bit string $\mathbf{z}$ of length $m = O(\log n)$ (we call $\mathbf{z}$ the encoding of queried index, or simply query), and splits $\mathbf{z}$ to two *additive* shares in $\mathbb{F}_2^m$, $\mathbf{z}_1$ and $\mathbf{z}_2$ (we call them sub-queries), and then sends them to the two servers respectively.

We construct PIR in the shuffle model based on this protocol. Abstractly, each client generates two sub-queries (or shares) $\mathbf{z}_1$ and $\mathbf{z}_2$ as if it was querying using the above two-server scheme but in fact sends both sub-queries to a single server through an anonymous channel (which shuffles the sub-queries together with that from many other clients). Observe that this is exactly an instance of secure aggregation in the split-and-mix approach [IKOS06, BBGN20, GMPV20], where each input is split into two shares; the hope is that the server would learn nothing given the shuffled encoding shares from many clients.

There are two issues with this toy protocol. The first issue is obvious—the server learns the sum of all the encoding strings, and therefore can easily distinguish two sets of query indices by comparing the sum of their shares and the sum of their encodings. Note that leaking the sum to the server is exactly the goal of secure aggregation, but the sum should not be leaked in the PIR context. This leakage can be easily eliminated by letting one of the clients add a dummy share (a random string) to hide the sum. The second issue is more involved. In fact, splitting each input into only two shares is not enough to guarantee security; this can be demonstrated through a simple counter-example: suppose that the server wishes to distinguish between the 2-additive shares of zeros and that of ones (sharing over $\mathbb{F}_2$) . In the latter case, there is always an equal number of ones and zeros in the shares, while this is not true for the former case. This approach can be generalized to a "counting" based strategy (for sharing over any Abelian groups) and allows for generic efficient distinguishing attacks (Appendix A). While splitting into more additive shares, e.g., 4, is sufficient [BBGN20], this means we need a 4-server PIR (that has additive sub-queries) and thus leads to worse communication—$O(n^{3/4})$ in the 4-server scheme compared to $O(n^{1/2})$ in the two-server scheme (Section 4.1). On the road map to our general protocol with $O(n^\gamma)$ communication (for any $\gamma > 0$), the first checkpoint is to bypass the above attack and achieve a protocol with $O(n^{1/2})$ communication; it turns out that the key ideas used for this also play a pivotal role in our final protocol design.

Randomizing inputs via the inner-outer paradigm. The core reason why the simple split-and-mix approach does not work with two additive shares is the presence of arbitrary correlation among the queries; indeed, if all queries were independent and uniformly random, then using two shares works perfectly. Our key insight to navigate around this is to randomize the queries using another PIR, resulting in *uniform random but pairwise independent* queries which is later shown to be sufficient for security.

Our construction employs a novel approach—the *inner-outer* paradigm, which composes a k-server PIR protocol as an *outer* layer with the previous 2-server PIR protocol (with 2-additive shares) as the inner layer. At a high level, the outer layer PIR randomizes the client queries before they get processed through the inner layer PIR. Below we call the outer layer protocol as OPIR and the inner layer protocol as IPIR.

Formally, the composition works as follows: for any database $x \in \{0,1\}^n$, on input an arbitrary query index $i \in [n]$, the client first runs the OPIR query algorithm to generate k queries $q_1, \ldots, q_k$; note that they naturally satisfy pairwise independence and each is uniformly random in the OPIR query space $\mathcal{Q}$, simply because of the security property of any PIR. Instead of sending them directly to the server, these queries are *interpreted as indices* to a new database x' of size $|\mathcal{Q}|$, where x' consists of the answers to all the possible OPIR queries (i.e., elements in $\mathcal{Q}$). Now the client runs IPIR query algorithm on the each of the k "indices" in $\{1, 2, \ldots, |\mathcal{Q}|\}$, and sends the IPIR sub-queries to the server. Specifically, the client maps an index to its encoding in the two-server protocol, and splits the encoding into 2 additive shares (sub-queries) in $\mathbb{F}_2^m$ where $m = O(\log |\mathcal{Q}|)$. Finally, to have the compilation work, the server needs to build the database x' for IPIR in advance, which is feasible as long as $|\mathcal{Q}|$ is polynomial in n.

The upshot of this compilation is that the server now sees a set of shuffled shares generated from uniformly random and pairwise independent query indices to the database x'. As we shall show next, this randomization achieves that, for any two multi-sets of queried indices I, I' with distance at most δ, the resulting multi-sets after processing through OPIR will be J, J' will have distance in expectation $\sqrt{\delta}$, even though J, J' are larger than I, I'. The distance further decreases to $\sqrt[4]{\delta}$ after processing through IPIR (additive sharing). We will show that having each client add only one random noise sub-query (on top of its real sub-queries) is sufficient to hide the $\sqrt[4]{\delta}$ distance from the server.

Analyzing split-and-mix with pairwise independence. We now prove that the split-and-mix approach is sufficient once we have pairwise independent queries from the OPIR; we will use a ball-and-bins formulation for this analysis. The full details are given in Section 6.1 and Appendix C.

Concretely, consider two arbitrary sets of client queries I and I'. Recall that each client query is first split into k uniformly random and pairwise-independent OPIR sub-queries; this is followed by 2-additive IPIR sharing, where each OPIR sub-query is further additively split into 2 shares in the IPIR space. Observe now that the OPIR queries of all clients can be viewed as throwing kC balls into $|\mathcal{Q}|$ bins where C is the number of clients and $\mathcal{Q}$ is the OPIR query space. Let $\mathcal{Y}(I)$ denote the distribution of the *balls-and-bins configuration* (i.e., a vector of random variables denoting the number of balls in each bin) of the OPIR sub-queries resultant from I. Next, given some configuration $\mathsf{y} \sim \mathcal{Y}(I)$, observe further that the 2-additive IPIR sharing can be viewed as creating a new, balls-and-bins configuration $\widetilde{\mathsf{y}}$ (now with $2kC$ balls), where each previous ball is now *split* into two balls; in particular, a ball in bin b within y results in two balls in random bins u and $b - u$ (where the bin labels are viewed as the additive group given by the IPIR query space). Denote the distribution of this new configuration by $\widetilde{\mathcal{Y}}(I)$. Roughly, the goal now is to prove that for arbitrary I and I', we can bound $\mathsf{SD}(\widetilde{\mathcal{Y}}(I), \widetilde{\mathcal{Y}}(I'))$ with some inverse polynomial in the number of clients C. This would imply that for an appropriately large $C = \mathsf{poly}(n)$, a server even with unbounded computation cannot distinguish between two arbitrary sets of client PIR queries, except with probability that is inverse polynomial in n.

Looking ahead however, we will require some extra "noise" balls to be added uniformly at random, essentially to "smooth out" the distribution of $\widetilde{\mathsf{y}}$; in the PIR context, this corresponds to client sending an additional random IPIR share. Denote the balls-and-bins distribution of the shares with noise added as $\widetilde{\mathcal{Y}}^*(I)$. Our proof proceeds in the following three major steps:

For our first proof step, we focus on the OPIR and bound the expected *distance* in the balls-and-bins configuration for any two $\mathcal{Y}(I)$ and $\mathcal{Y}(I')$—intuitively, the distance here captures how many balls must be moved in one configuration to make it identical to the other. We show (Lemma 6.2), that the expected distance between the two configurations is bounded by $\sqrt{kC \cdot |\mathcal{Q}|/2}$. In other words, although I and I' can differ in the queries of all C clients, the expected distance between their OPIR queries is proportional to $\sqrt{C}$.

Now, for our second proof step, we analyze the additive splitting which takes place through the IPIR. Consider, in particular, two configurations y_1 and y_2 for the OPIR sub-queries. First, we show that it is sufficient to look only at the places where y_1 and y_2 differ in the context of the final statistical distance (see Lemma C.1); this allows us to on expectation, focus only on roughly $\sqrt{C}$ balls from the OPIR configurations when we later look at the IPIR sharing. We now show that when y_1 and y_2 have distance δ, post additive-sharing in the IPIR , the distance between the corresponding $\widetilde{\mathsf{y}}_1$ and $\widetilde{\mathsf{y}}_2$ reduces to $\Theta(\sqrt{\delta})$ (see Lemma C.2). Consequently, combining this with the first proof step implies that any sets of original client indices, once put through both the OPIR and IPIR, will have distance on expectation proportional to $\sqrt[4]{C}$.

The third and final proof step now shows that adding just 1 noise query per client results in being able to "hide" this $\sqrt[4]{C}$ difference in order to get $1/\mathsf{poly}(n)$ security; adding more noise queries improves the asymptotic bound on the number of clients needed to achieve the same security level. The analysis in this step roughly models the "toy in sand" problem—intuitively, how much "sand" (i.e., noise balls or queries) are needed to hide which bin a "toy" ball was initially put in.

Combining all the steps, we can show $\epsilon(n)$ statistical security as the total number of clients C is at least $\Omega(n^5/\epsilon^8)$; consequently, we get can any inverse-polynomial security where C is also polynomial in n. The (per-client) communication complexity for this construction is $O(n^{1/2})$.

Improving communication using CNF-shares. Following this, in Section 6.2, we show how a CNF-sharing based construction can be used as the IPIR to reduce the communication complexity; in particular, using an s-CNF sharing allows us to reduce the communication cost to $O(n^{1/s})$ given $\Omega(n^{2s+1}/\epsilon^8)$ clients for statistical security ϵ. This cleanly generalizes our earlier construction.

The security proof follows a similar outline as before but is somewhat more involved. We find a nice group theoretic formulation of the problem of understanding the symmetries within the CNF-sharing, which allows us to greatly simplify the analysis by leveraging simple results from that domain.

Lower bound on security. We show a lower bound on security for protocols within our inner-outer paradigm, by showing that negligible statistical distance cannot be achieved in this realm. To prove this, we borrow an idea from Ghazi et.al [GMPV20, Theorem 6], and extend their results on secret sharing to the PIR context. We observe that the query algorithms of multi-server PIR protocols can be viewed as secret sharing; this allows us to show that if the total number of possible ways to secret share a query index is $K = p_1(n)$ and there are $C = p_2(n)$ clients, then there must exist two sets of input indices with some $1/p_3(n)$ statistical distance, where p_1, p_2, p_3 are all polynomials in n.

3 Related Work

Below we survey existing work in the standard PIR model and contrast them with this work.

Single-server PIR. Kushilevitz and Ostrovsky [KO97] gave the first single-server PIR scheme based on quadratic residuosity assumption with linear server computation. Beimel, Ishai and Malkin [BIM00] later showed that linear server computation (no matter in multi-server or single-server schemes) is inherent if no extra storage (at the server or the clients) is allowed. Subsequently, there are many works aiming to construct PIR with fast server computation under different models, and we categorize them as follows.

PIR with batched queries. "Batch PIR" processes a batch of queries to achieve sublinear cost per query. The batch PIR schemes [IKOS04, Hen16, CHLR18, ACLS18, MR23] work in the standard PIR model but amortize costs when a *single* client wants to simultaneously query multiple records, and they require the client to know the sequence of queries it makes in advance (i.e., the batch is non-adaptive). In contrast, we consider a fundamentally different model, where multiple clients simultaneously query a single server and their queries are shuffled. A qualitative advantage of the shuffle model is that it enables sublinear computation in the single-server setting without using cryptographic assumptions.

PIR with preprocessing. This class of schemes [BIM00, WY05, CHR17, BIPW17, Lip09, LMW23] utilize extra storage at the server(s), where the database is encoded into some (larger) forms that allow answering each query with sublinear computation. Our schemes fall in this category, where the server pre-computes the answers to all the sub-queries and stores them in a table, and the client performs a sublinear number of table lookups (in a private way). This is feasible (polynomial time in n) in terms of preprocessing cost and server storage when each sub-query has bit length $O(\log n)$.

PIR in the offline/online model. Recent work [CGHK22] construct PIR schemes where the extra storage is incurred at the clients: a client first runs an offline phase with the server, where the server does a (super)linear computation to generate hints and sends them to the client. In the subsequent online phase, the server answers each query with sublinear cost and the client obtains the queried item with the help of the hints. The hints can be reused for $n^{1/4}$ number of online queries, namely, the offline cost is amortized over an adaptive batch of queries.

There are also schemes [HHCG$^+$23, DPC23, ZPSZ23] that work in a slightly different offline/online model, where they allow $\mathsf{poly}(n)$ number of online queries (from the same client) but with linear (though concretely fast) server computation per query [HHCG$^+$23, DPC23]; or they have linear client communication in the offline phase in exchange for sublinear online computation [ZPSZ23]. Our schemes do not require extra hints at clients, and the client can make an unlimited number of queries after a one-time server preprocessing.

Other works towards doubly efficient PIR. Ishai et al. [IKOS06] give a single-server PIR construction in the shuffle model as follows: each client runs a query algorithm of an information-theoretic multi-server PIR [BIK05] on its query index and generates sub-queries, following which the sub-queries from many clients are shuffled together and sent to the server. However, this construction is shown to be secure only under non-standard computational assumptions (specifically the hardness of reconstructing noisy low-degree curves in a low-dimensional space [IKOS06, CS03]). In contrast, our construction provides statistical security. The trade-off is that we require more clients accessing the database compared to theirs, and settle for inverse-polynomial security error. We leave open the possibility of removing these limitations. An alternative avenue for future work is obtaining more efficient *computational* PIR schemes in the shuffle model, improving the efficiency of the shuffle-based protocol from [IKOS06]. While there are single-server doubly efficient computational PIR protocols even in the plain model [BIPW17, CHR17, LMW23], they either rely on obfuscation or fully homomorphic encryption and therefore the potential of achieving good concrete efficiency along this direction might be limited. The shuffle model seems to have better potential for practical single-server solutions.

Differential privacy (DP) for PIR. A line of work [TDG16, AIVG22] considers the DP notion for PIR assuming client anonymity. Here, clients send their query indices via onion routing to the server, and privacy is guaranteed by the shuffling of client indices along with some noise queries. Here DP guarantees that the server cannot distinguish neighboring sets of queries (i.e., differing in exactly one client). Unfortunately, DP is substantially *weaker* than standard PIR security and therefore insufficient in any application where client queries can be *arbitrarily correlated*, as we illustrate below.

Consider an example where a disproportionately large number of clients access the same sensitive entry in the database; in standard security notion, this should be indistinguishable (statistically or computationally) from the case where nobody accesses this entry, whereas the DP solutions fail here. Meanwhile, in many applications, the information of whether clients are correlated accessing the same or similar items can give the adversaries extra powers: consider a cloud service that stores encrypted medical records, and the adversary can use the access frequencies of the records together with public health statistics to recover the underlying plaintext of the encrypted records [ZKP16, GKL+20]. In contrast, our construction achieves a stronger notion: for any set of query indices, the messages observed by the server look close to random.

The "Split and mix" technique. A core idea in our construction follows from an ingenious split-and-mix approach for secure summation by Ishai et al. [IKOS06]. Specifically, they give a one-round single-server secure aggregation protocol as follows: Each client *splits* its input into k additive shares; then, as part of the shuffle model, these shares from all the C clients are *mixed* together before being sent to the aggregation server who simply outputs the sum of all the shares. The security goal here is that server cannot infer anything about a particular client's input. More precisely, the shuffled shares of any two tuples of client inputs (with equal sum) should look indistinguishable. Ishai et al. [IKOS06] show that statistical security of $2^{-\sigma}$ can be achieved by using per-input $k = \Theta(\log C + \log p + \sigma)$ additive shares over a group of size p. Recent works [BBGN20, GMPV20] improve this bound to $k = \lceil 2 + \frac{2\sigma + \log_2(p)}{\log_2(C)} \rceil$ and show that at least 4 shares are necessary.

In our shuffle PIR context, we find that 2 additive shares are sufficient due to our query randomization technique and the usage of additional *noise* queries; this cannot be done in the summation setting as the final output could change. Towards reducing the communication of our PIR protocol, we also generalize the split-and-mix approach to CNF shares.

4 Preliminaries

Basic notation. For $n \in \mathbb{N}$, we use $[n]$ to denote the set $\{1, 2, \ldots, n\}$. $\mathbb{F}$ denotes a finite field. $\mathfrak{S}_c$ denotes the symmetric group containing all permutations of c elements. We use bold letters to denote vectors (e.g., $\mathbf{z}$). We use $\mathsf{SD}(\mathcal{D}_1, \mathcal{D}_2)$ to denote the statistical distance between the distributions $\mathcal{D}_1$ and $\mathcal{D}_2$.

Unless specified, logarithms are taken to the base 2. The notation $\mathsf{poly}(\cdot)$ refers to a fixed but unspecified polynomial in its parameter; we use $\mathsf{polylog}(\cdot)$ to mean $\mathsf{poly}(\log(\cdot))$. The notation $\widetilde{O}$ hides arbitrary polylogarithmic factors, i.e., $f(n) = \widetilde{O}(g(n))$ if $f(n) = O(g(n)) \cdot \mathsf{polylog} n$.

We use $\xrightarrow{\$}$ to denote uniformly random sampling, $\rightarrow$ for output by deterministic algorithms, and $\ {\scriptstyle\$}\!\!\rightarrow$ for output by randomized algorithms.

4.1 Multi-Server Information-Theoretic PIR

We now introduce the standard notion of multi-server information-theoretic PIR. We start with the basic definition below.

Definition 4.1 (PIR)**.** Let Σ be a finite alphabet. A k-server PIR protocol over Σ is a tuple $\Phi = (\mathsf{Setup}, \mathsf{Query}, \mathsf{Answer}, \mathsf{Recon})$ with the following syntax:

- $\mathsf{Setup}(x) \rightarrow P_x$: a deterministic algorithm executed by all servers that takes in an n-entry database $x \in \Sigma^n$ and outputs its encoding P_x.
- $\mathsf{Query}(i; n) \ {\scriptstyle\$}\!\!\rightarrow ((q_1, \ldots, q_k), \mathsf{st})$: a randomized algorithm (parameterized by n) executed by the client that takes in an index $i \in [n]$, and outputs sub-queries $q_1, \ldots, q_k$ and a state st. The sub-query q_ℓ is sent to the ℓ-th server.
- $\mathsf{Answer}^\ell(P_x, q_\ell) \rightarrow a_\ell$: a deterministic algorithm executed by the ℓ-th server that takes in the encoding P_x and a sub-query q_ℓ, and outputs an answer a_ℓ. Since the Answer algorithm may be different for different servers, we use ℓ to denote the algorithm used by server ℓ.
- $\mathsf{Recon}((a_1, \ldots, a_k), \mathsf{st}) \rightarrow x_i$: a deterministic algorithm executed by the client that takes in answers $a_1, \ldots, a_k$ (where a_ℓ is from the ℓ-th server) and the state st, and outputs $x_i \in \Sigma$.

Φ needs to satisfy the following correctness and security properties:
Correctness. For all $n \in \mathbb{N}$, any database $x = (x_1, \ldots, x_n) \in \Sigma^n$, and all $i \in [n]$,

$$\Pr\left[\ \mathsf{Recon}((a_1, \ldots, a_k), \mathsf{st}) = x_i : \begin{array}{rcl} P_x & \leftarrow & \mathsf{Setup}(x) \\ ((q_1, \ldots, q_k), \mathsf{st}) & \leftarrow\!{\scriptstyle\$} & \mathsf{Query}(i; n) \\ (a_1, \ldots, a_k) & \leftarrow & (\mathsf{Answer}^\ell(P_x, q_\ell))_{\ell=1}^k \end{array}\right] = 1.$$

Intuitively, correctness says that the client always gets the correct value of x_i.

Security. For all $n \in \mathbb{N}$, $i \in [n]$, and $T \subset [k]$, define the distribution

$$\mathcal{D}_n(i, T) := \left\{ \{q_\ell\}_{\ell \in T} : ((q_1, \ldots, q_k), \mathsf{st}) \leftarrow\!{\scriptstyle\$}\ \mathsf{Query}(i; n) \right\}.$$

We say that Φ has (t, ϵ)-privacy (where $t < k$, and $\epsilon = \epsilon(n)$), if for all $n \in \mathbb{N}$, any two indices $i, i' \in [n]$, and any set $T \subset [k]$ such that $|T| < t$, we have

$$\mathsf{SD}(\mathcal{D}_n(i, T), \mathcal{D}_n(i', T)) \leq \epsilon(n).$$

Intuitively, (t, ϵ)-privacy says that any set of less than t colluding servers has a distinguishing advantage at most ϵ.

We now provide as background common PIR schemes that will be important later for our construction for PIR in the shuffle model. The constructions employ the following general outline: The servers encode the database $x \in \Sigma^n$ as a polynomial P_x. To query the database at position i, the client first encodes i into a vector $\mathbf{z}^{(i)}$ where the encoding is defined in a way that results in $P_x(\mathbf{z}^{(i)}) = x_i$. The client now evaluates P_x at $\mathbf{z}^{(i)}$ while hiding $\mathbf{z}^{(i)}$ from the servers: it secret shares $\mathbf{z}^{(i)}$ into k shares, and each share is sent to one of the k servers (through e.g., additive or Shamir sharing). Each server can then evaluate P_x on one share and send the result to the client, who is able to reconstruct the entry x_i.

4.1.1 Two-Server PIR with Additive Shares

The first construction we describe is a PIR scheme from Beimel et al. [BIK05] which uses two non-colluding servers. Figure 4.1 contains the full description.

<u>*Setup.*</u> Consider a field $\mathbb{F}$ within which Σ can be encoded. The Setup algorithm encodes a database $x \in \Sigma^n$ into an m-variate polynomial $P_x \in \mathbb{F}[Z_1, \ldots, Z_m]$ as follows. First, choose m and $d < m$ such that $\binom{m}{d} \geq n$, and let $M = (M_1, \ldots M_n)$ denote a list of n monomials in the variables $Z_1, \ldots, Z_m$ with total degree exactly d and the degree of each variable at most 1. For simplicity, we pick the first n such monomials in lexicographic order of the variable indices (e.g., $Z_1 Z_2 Z_3$ appears before $Z_1 Z_2 Z_4$). The encoding P_x is now simply the linear combination $P_x = \sum_{i=1}^{n} x_i M_i$.[1]

<u>*Query.*</u> The Query algorithm starts by encoding the query index $i \in [n]$ into a binary vector $\mathbf{z}^{(i)} = (z_1^{(i)}, \ldots, z_m^{(i)}) \in \{0,1\}^m$ defined such that each $z_j^{(i)} = 1$ if and only if the monomial M_i contains the variable Z_j. Observe here that the Hamming weight of $\mathbf{z}^{(i)}$ is d since the monomials are also of degree d. Such encoding ensures that $P_x(\mathbf{z}_i) = x_i$. Then the sub-queries are generated by splitting $\mathbf{z}^{(i)}$ into two additive shares $\mathbf{z}_1^{(i)} = (z_{1,1}^{(i)}, \ldots, z_{m,1}^{(i)})$ and $\mathbf{z}_2^{(i)} = (z_{1,2}^{(i)}, \ldots, z_{m,2}^{(i)})$, i.e., $\mathbf{z}^{(i)} = \mathbf{z}_1^{(i)} + \mathbf{z}_2^{(i)}$. Here, $\mathbf{z}_\ell^{(i)}$ is sent to the ℓ-th server for $\ell = 1, 2$.

<u>*Answer.*</u> The Answer^ℓ algorithm run by the servers first views the database encoding P_x as a $2m$-variate polynomial P'_x defined as:

$$P'_x(Z_{1,1}, Z_{1,2}, \ldots, Z_{m,1}, Z_{m,2}) = P_x(Z_{1,1} + Z_{1,2}, \ldots, Z_{m,1} + Z_{m,2}).$$

Now, the ℓth server selects all the monomial terms in P'_x such that the number of $Z_{\_,\ell}$ (i.e., the variables where the second subscript is ℓ) is at least half of the variables in that term (in the exactly half case, the monomials are split between the two servers in a pre-determined way). Note that the total number of monomials in P'_x is $2^d \cdot n$, so there should be $2^{d-1} \cdot n$ monomials for each server. The ℓ-th server then evaluates its selected monomials at the point $\mathbf{z}_\ell^{(i)}$ and responds with the sum as the answer a_ℓ (which is now a polynomial in the remaining m variables). Further, observe that each monomial in P'_x is of degree d, and so after the server evaluation, the answer polynomial a_ℓ will be of degree at most $d/2$.

<u>*Reconstruction.*</u> Finally, given answer polynomials a_1, a_2, the client evaluates a_1 at $\mathbf{z}_2^{(i)}$ and a_2 at $\mathbf{z}_1^{(i)}$, and sums up the evaluation results in $\mathbb{F}$ to get $P_x(\mathbf{z}^{(i)}) = x_i$.

[1]One can choose a more complicated encoding in [BIK05] (E1 encoding scheme) that allows better parameters, namely $\sum_{\ell=0}^{d} \binom{m}{\ell} \geq n$.

Let x be a database with size n and $\mathbb{F}$ be a field, where each entry x_i is in $\Sigma = \mathbb{F}$.

- $\mathsf{PIR.Setup}(x) \to P$:
  1. Choose m, d such that $\binom{m}{d} \geq n$.
  2. Let $M = (M_1, \ldots, M_n)$ be a list of n monomials in $\mathbb{F}[Z_1, \ldots, Z_m]$ with total degree d and intermediate degree at most 1. Sort all monomials that have m variables with degree d by a lexicographic order of the variables indices.
  3. Compute $P_x = \sum_{i=1}^{n} x_i M_i \in \mathbb{F}[Z_1, \ldots, Z_m]$.
  4. Compute a $2m$-variate degree-d polynomial P from P_x such that $P(Z_{1,1}, Z_{1,2}, \ldots, Z_{m,1}, Z_{m,2}) = P_x(Z_{1,1} + Z_{1,2}, \ldots, Z_{m,1} + Z_{m,2})$.
  5. Output P.
- $\mathsf{PIR.Query}(i; n) \to ((q_1, q_2), \mathsf{st})$, where $i \in [n]$:
  1. Let $\mathbf{z} = (z_1, \ldots, z_m)$ be the i-th binary vector such that $z_j = 1$ if and only if the monomial M_i contains the variable Z_j.
  2. Let $\mathbf{z}_1 \overset{\$}{\leftarrow} \mathbb{F}_2^m$, $\mathbf{z}_2 \leftarrow \mathbf{z} - \mathbf{z}_1$; and let $q_\ell \leftarrow \mathbf{z}_\ell$ for $\ell = 1, 2$. Set $\mathsf{st} = (\mathbf{z}_1, \mathbf{z}_2)$.
  3. Output $((q_1, q_2), \mathsf{st})$.
- $\mathsf{PIR.Answer}^\ell(P, q_\ell) \to a_\ell$ (for $\ell = 1, 2$):
  1. Let $\{M'_j\}_{j \in [2^m n]}$ be all monomials where the number $Z_{\_,\ell}$ is at least half of the variables.
  2. Output $a_\ell \leftarrow \sum_{j \in [2^m n]} M'_j(q_\ell)$.
- $\mathsf{PIR.Recon}((a_1, a_2), \mathsf{st}) \to x_i$:
  1. Parse st as $(\mathbf{z}_1, \mathbf{z}_2)$.
  2. Compute $x_i \leftarrow a_1(\mathbf{z}_2) + a_2(\mathbf{z}_1)$ (note that a_1 and a_2 are polynomials).
  3. Output x_i.

Construction 4.1: A two-server information-theoretic PIR [BIK05].

Cost. The parameters m and d can be chosen to be both $\Theta(\log n)$ such that $\binom{m}{d} \geq n$. In this case, the query size is $O(\log n)$ (since m elements in $\mathbb{F}_2$ are sent to each server) and the answer size is $O(\sqrt{n})$ (since specifying an m-variate polynomial of degree $d/2$ requires $\binom{m}{d/2} = O(\sqrt{n})$ terms).

k-server PIR with additive shares. The above protocol can also be generalized to k servers where the encoding $\mathbf{z}$ is now split into k additive shares. In this case, the servers express the m-variate degree-d polynomial P_x as km-variate degree-d polynomial P'_x. That is,

$$P'_x(Z_{1,1}, \ldots, Z_{1,k}, \ldots, Z_{m,1}, \ldots, Z_{m,k}) = P_x(Z_{1,1} + \ldots + Z_{1,k}, \ldots, Z_{m,1} + Z_{m,k}).$$

Let $\mathcal{Z}_\ell$ be the set of monomials such that for each monomial, there are more $Z_{\_,\ell}$ than $Z_{\_,\ell'}$ for any $\ell' \neq \ell$. The set $\mathcal{Z}_\ell$ is assigned to the ℓ-th server. Moreover, the monomials in P'_x but not in any of $\mathcal{Z}_{\_,\ell}$'s will be divided to k servers in a pre-determined way. To issue a query for index i, the client encodes it as before to a binary string $\mathbf{z} \in \mathbb{F}_2^m$, and then splits it to k additive shares over $\mathbb{F}_2^m$, denoted as $\mathbf{z}_1, \ldots, \mathbf{z}_k$. The client sends to the ℓ-th server the share $\mathbf{z}_\ell$, and the server evaluates

Let x be a database with size n, each entry x_i is in $\Sigma = \mathbb{F}$. There are s non-colluding servers.

- $\mathsf{PIR.Setup}(x) \to P$:
  1. Choose m, d such that $\binom{m}{d} \geq n$.
  2. Let $M = (M_1, \ldots, M_n)$ be a list of n monomials in $\mathbb{F}[Z_1, \ldots, Z_m]$ with total degree exactly d and intermediate degree at most 1. Sort all monomials that have m variables with degree d by a lexicographic order of the variables indices.
  3. Compute $P_x = \sum_{i=1}^{n} x_i M_i \in \mathbb{F}[Z_1, \ldots, Z_m]$.
  4. Compute a sm-variate degree-d polynomial P from P_x such that $P(Z_{1,1}, \ldots, Z_{1,s}, \ldots, Z_{m,1} \ldots Z_{m,s}) = P_x(Z_{1,1} + \ldots + Z_{1,s}, \ldots, Z_{m,1} + \ldots + Z_{m,s})$.
  5. Output P.
- $\mathsf{PIR.Query}(i; n) \to ((q_1, \ldots, q_s), \mathsf{st})$, where $i \in [n]$:
  1. Let $\mathbf{z} = (z_1, \ldots, z_m)$ be the i-th binary vector such that $z_j = 1$ if and only if the monomial M_i contains the variable Z_j.
  2. Let $\mathbf{z}_1, \ldots, \mathbf{z}_{s-1} \overset{\$}{\leftarrow} \mathbb{F}_2^m$ and $\mathbf{z}_s \leftarrow \mathbf{z} - \sum_{j=1}^{s-1} \mathbf{z}_j$.
  3. Let $q_\ell \leftarrow (\mathbf{z}_{\ell+1}, \ldots, \mathbf{z}_s, \mathbf{z}_1, \ldots, \mathbf{z}_{\ell-1})$ for $\ell \in [s]$. // cyclic shift
  4. Set $\mathsf{st} = (\mathbf{z}_1, \ldots, \mathbf{z}_s)$.
  5. Output $((q_1, \ldots, q_s), \mathsf{st})$.
- $\mathsf{PIR.Answer}^\ell(P, q_\ell) \to a$, for $\ell \in [s]$:
  1. Let $\{M'_j\}_{j \in [s^m n]}$ be all monomials pre-determined such that the number of $Z_{\_,\ell}$ is at most $1/s$ fraction.
  2. Output $a \leftarrow \sum_{j \in [s^d n]} M'_j(q_\ell)$.
- $\mathsf{PIR.Recon}((a_1, \ldots, a_\ell), \mathsf{st}) \to x_i$:
  1. Parse st as $(\mathbf{z}_1, \ldots, \mathbf{z}_s)$.
  2. Compute $x_i \leftarrow \sum_{\ell \in [s]} a_\ell(\mathbf{z}_1, \ldots, \mathbf{z}_\ell - 1, \mathbf{z}_{\ell+1}, \mathbf{z}_s)$.
  3. Output x_i.

Construction 4.2: An s-server PIR with CNF shares [BIK05]. Note that when $s = 2$, this is simply the 2-server additive PIR.

the assigned monomials using $\mathbf{z}_\ell$. The evaluation result is a polynomial of degree $(k-1)d/k$; this implies the answer size (which dominates the communication cost) is $O(n^{(k-1)/k})$.

Observe that using more additive shares gives worse efficiency but better privacy (since collusion between any $k-1$ servers can be tolerated). Efficiency can be significantly improved to $O(n^{1/k})$ using CNF shares [ISN87] (instead of additive shares) where each server is now given a different $(k-1)$-sized subset of the additive shares. This is because the evaluation of P_x at $k-1$ shares results in an answer polynomial of degree at most $O(n^{1/k})$. The efficiency gain, however, comes at the cost of much stronger non-collusion assumption for PIR, namely that no two database servers can collude. Looking ahead, an interesting consequence of using the shuffle model is that our CNF-sharing based construction (Section 6.2) can significantly reduce communication *without* making any non-collusion assumptions on database servers (since there is only one database).

For simplicity, going forward, we will refer to the k-server PIR with additive shares as k-*additive* PIR and its CNF-variant as k-CNF PIR; we describe this in Figure 4.2.

4.1.2 k-Server PIR with Shamir Shares

In this section, we describe the k-server t-private PIR that uses Shamir secret sharing from [BIK05]. Full description is provided in Figure 4.3. We also call this the Reed-Muller PIR as it is closely related to Reed-Muller code.

```
Let $x = (x_1, \ldots, x_n) \in \mathbb{F}^n$ be a database.

    $\mathsf{PIR.Setup}(x) \to P_x$:
        1. Choose parameters $m, d, k, t$ such that
           $\binom{m+d}{d} \geq n$ and $|\mathbb{F}| > k > td$.
        2. Compute $P_x = \sum_{i=1}^{n} x_i P^{(i)}(z_1, \ldots, z_m)$, where $P^{(i)}(\mathsf{PIR.Enc}(i)) = 1$ and $P^{(i)}(\mathsf{PIR.Enc}(j)) = 0$
           for all $i, j \in [n]$ and $i \neq j$.
        3. Output $P_x$.
    $\mathsf{PIR.Query}(i; n) \to ((q_1, \ldots, q_k), \mathsf{st})$, where $i \in [n]$:
        1. Run $\mathsf{PIR.Enc}(i)$ and gets $\mathbf{z} \in \mathbb{F}^m$.
        2. Choose a set of degree-$t$ random polynomials $R = (R_1, \ldots, R_m)$ such that $R(\mathbf{0}) = \mathbf{z}$.
        3. For $\ell \in [k]$:
            – Randomly choose $r_\ell$ from $\mathbb{F}$.
            – Set $q_\ell \leftarrow Q(r_\ell)$. Note that each $q_\ell \in \mathbb{F}^m$.
        4. Set $\mathsf{st} = (r_1, \ldots, r_k)$.
        5. Output $((q_1, \ldots, q_k), \mathsf{st})$.
    $\mathsf{PIR.Answer}(P_x, q) \to a$:
        1. Compute $a \leftarrow P_x(a)$.
        2. Output $a$.
    $\mathsf{PIR.Recon}((a_1, \ldots, a_k), \mathsf{st}) \to x_i$:
        1. Parse $\mathsf{st} = (r_1, \ldots, r_k)$.
        2. Interpolate a degree-$td$ univariate polynomial $R \circ P_x$ from $\{(r_\ell, a_\ell)\}_{\ell=1}^{k}$.
        3. Output $x_i \leftarrow (R \circ P_x)(0)$.
```

Construction 4.3: A k-server t-private PIR based on Reed-Muller code [BIK05].

Setup. Consider a field $\mathbb{F}$ within which Σ can be encoded. The Setup algorithm encodes a database $x \in \Sigma^n$ into a polynomial $P_x \in \mathbb{F}[Z_1, \ldots, Z_m]$ as follows: First, choose m and d such that $\binom{m+d}{d} \geq n$ and $|\mathbb{F}| > k > td$ (typically m, d, t are chosen first and then k and the field size $|\mathbb{F}|$ are deteremined accordingly). Let $\alpha_0, \ldots, \alpha_d$ be distinct elements in $\mathbb{F}$ (note that $d < |\mathbb{F}|$). The index i is encoded to the i-th vector $\mathbf{z}^{(i)}$ of the form $(\alpha_{\lambda_1}, \ldots, \alpha_{\lambda_m}) \in \mathbb{F}^m$ where $\sum_{j=1}^{m} \lambda_j \leq d$. There exists a set of polynomials $P^{(i)}(z_1, \ldots, z_m)$ of degree at most d such that $P^{(i)}(\mathbf{z}^{(i)}) = 1$ and $P^{(i)}(\mathbf{z}^{(j)}) = 0$ for all $i, j \in [n]$ and $i \neq j$. The full details of this encoding and the construction of $P^{(i)}$'s are provided in [BIK05, Appendix B].

<u>*Query.*</u> To generate the sub-queries, after encoding the index i to $\mathbf{z}^{(i)}$, the client first chooses m univariate polynomials $(R_1, \ldots, R_m) = R$ each of degree t such that $R(\mathbf{0}) = (R_1(0), \ldots, R_m(0)) = \mathbf{z}^{(i)}$. It then randomly picks $r_1, \ldots, r_k \in \mathbb{F}$ and computes the sub-query to be sent to the ℓ^{th} server as $q_\ell = R(r_\ell) \in \mathbb{F}^m$.

<u>*Answer.*</u> The Answer^ℓ algorithm evaluates P_x at q_ℓ and sends back $a_\ell = P_x(q_\ell)$. Note that the answer algorithm for this protocol is the same for all k servers.

<u>*Reconstruction.*</u> Finally, the Recon algorithm uses Lagrange interpolation on the points $(r_1, a_1), \ldots, (r_k, a_k)$ to compute a degree td polynomial $S = P_x \circ R$; the evaluation $S(0)$ will give the desired database entry x_i. This interpolation is possible when $k > td$ and $|\mathbb{F}| > k$.

Other notation. For a PIR protocol Φ, we use $\mathcal{E}_\Phi$ to denote the encoding space of all indices. We use $\mathcal{Q}_\Phi$ to denote the space of all possible sub-queries (note that $\mathcal{Q}_\Phi$ may not equal $\mathcal{E}_\Phi$). For example, in the two-server construction above, $\mathcal{E}_\Phi$ contains all binary strings with Hamming weight d, and the space $\mathcal{Q}_\Phi$ is $\mathbb{F}_2^m$, i.e, in this case $\mathcal{E}_\Phi \subset \mathcal{Q}_\Phi$.

4.2 Balls and Bins

We formulate the core analysis of our constructions using the widely-used balls-and-bins problem, which we provide background and notation for here. Abstractly, the balls-and-bins problem analyzes the distribution of B (identical) balls thrown into N bins according to some distribution D (often independent and uniformly at random). To denote a final configuration of balls, we use a N-length vector $\mathbf{u} = (u_0, \ldots, u_{N-1})$ where u_i denotes the number of balls in bin i. Since our analysis often deals with sharing over a group $\mathbb{G}$, we may also label the bins using elements from $\mathbb{G}$; when $\mathbb{G}$ is unspecified, it is taken to be $\mathbb{Z}_N$. In particular, we define the following:

Definition 4.2 (Valid configuration)**.** We say that a vector $\mathbf{u} = (u_0, \ldots, u_{N-1})$ is a valid (B, N) balls-and-bins configuration, or simply that $\mathbf{u}$ is (B, N)-valid if each $u_i \in \mathbb{Z}^{\geq 0}$ and $\sum_i u_i = B$.

Definition 4.3. Given (B, N)-valid configurations $\mathbf{u} = (u_0, \ldots, u_{N-1})$ and $\mathbf{v} = (v_0, \ldots, v_{N-1})$, we define the following useful terms:

- The *edit distance*, denoted by $\mathsf{ED}(\mathbf{u}, \mathbf{v})$ is defined as $\mathsf{ED}(\mathbf{u}, \mathbf{v}) = \frac{1}{2} \sum_{i=0}^{N-1} |u_i - v_i|$.

  Intuitively, this denotes the number of balls that need to be moved to convert $\mathbf{u}$ to $\mathbf{v}$. Note that the distance is symmetric since $\mathsf{ED}(\mathbf{u}, \mathbf{v}) = \mathsf{ED}(\mathbf{v}, \mathbf{u})$. The edit distance between two distributions $\mathcal{U}$ and $\mathcal{V}$, denoted by $\mathsf{ED}(\mathcal{U}, \mathcal{V})$; can now be defined as $\mathbb{E}_{\mathbf{u} \sim \mathcal{U}, \mathbf{v} \sim \mathcal{V}} [\mathsf{ED}(\mathbf{u}, \mathbf{v})]$.

- The *ball-intersection* $\mathbf{u} \sqcap \mathbf{v}$ is defined as$(c_0, \ldots, c_{N-1})$ where each $c_i = \min(u_i, v_i)$.

- The *ball-difference* $\mathbf{u} \ominus \mathbf{v}$ is defined as$(u'_0, \ldots, u'_{N-1})$ where each $u'_i = \max(0, u_i - v_i)$.

5 Single-Server PIR in the Shuffle Model: Definitions and Preliminary Results

We now formally define single-server PIR in the shuffle model. The setting here is to consider a single server but many query-making clients while still retaining information-theoretic security. Importantly, we do not assume any coordination among clients.

Definition 5.1 (PIR in the shuffle model)**.** Let Σ be a finite alphabet. A (single-server) PIR protocol (over Σ) in the shuffle model is a tuple $\mathsf{ShPIR} = (\mathsf{Setup}, \mathsf{Query}, \mathsf{Answer}, \mathsf{Recon})$ with a syntax similar to that of a k-server PIR (Definition 4.1) except for a few key changes. In particular:

- $\mathsf{Setup}(x) \to P_x$: a deterministic algorithm executed by the server that takes in an n-entry database $x \in \Sigma^n$ and outputs its encoding P_x.
- $\mathsf{Query}(i; n) \overset{\$}{\to} (q_1, \ldots, q_k)$: a randomized algorithm (parameterized by n) executed by the client that takes in an index $i \in [n]$, and outputs sub-queries $q_1, \ldots, q_k$. Unlike in Definition 4.1, k may be a function of n; this is possible since the shuffle model does not require k physical servers. Further, all sub-queries will be sent to the same server. For simplicity, here we omit the state in Definition 4.1.
- $\mathsf{Answer}(P_x, q_\ell) \to a_\ell$: a deterministic algorithm executed by the server that takes in the encoding P_x and a sub-query q_ℓ, and outputs an answer a_ℓ. Unlike in Definition 4.1, there is a single Answer algorithm.
- $\mathsf{Recon}(a_1, \ldots, a_k) \to x_i$: a deterministic algorithm executed by the client that takes in answers $a_1, \ldots, a_k$, where for all $\ell \in [k]$, a_ℓ is the answer to the client's sub-query q_ℓ; and outputs $x_i \in \Sigma$.

ShPIR needs to satisfy the following correctness property:

Correctness. For all $n \in \mathbb{N}$, database $x = (x_1, \ldots, x_n) \in \Sigma^n$, and $i \in [n]$,

$$\Pr\left[\mathsf{Recon}(a_1, \ldots, a_k) = x_i : \begin{array}{rcl} P_x & \leftarrow & \mathsf{Setup}(x) \\ (q_1, \ldots, q_k) & \overset{\$}{\leftarrow} & \mathsf{Query}(i; n) \\ (a_1, \ldots, a_k) & \leftarrow & (\mathsf{Answer}(P_x, q_\ell))_{\ell=1}^{k} \end{array} \right] = 1.$$

ShPIR also needs to satisfy the following security property in the model where client queries are shuffled before being sent to the server.

Security. We will parameterize security by a shuffler Π and a minimum number of honest client queries C. Formally, let $\Pi = \{\Pi_c\}_{c \in \mathbb{N}}$ be an ensemble such that Π_c is a distribution over the symmetric group $\mathfrak{S}_c$. When Π is unspecified, we assume that each Π_c is a uniform distribution over $\mathfrak{S}_c$; we refer to this as the uniform or perfect shuffler. We discuss imperfect shufflers in Appendix B.

For a given n, Π, and C, and given a tuple $I = (i_1, \ldots, i_C) \in [n]^C$ of client query indices, define the distribution

$$\widetilde{\mathcal{D}}_{n,\Pi,C}(I) = \left\{ \pi(\mathbf{q}) : \begin{array}{l} (q_1^{(1)}, \ldots, q_k^{(1)}) \overset{\$}{\leftarrow} \mathsf{Query}(i_1; n) \\ \cdots \\ (q_1^{(C)}, \ldots, q_k^{(C)}) \overset{\$}{\leftarrow} \mathsf{Query}(i_C; n) \\ \mathbf{q} \leftarrow (q_1^{(1)}, \ldots, q_k^{(1)}, \ldots, q_1^{(C)}, \ldots, q_k^{(C)}) \\ \pi \overset{\$}{\leftarrow} \Pi_{kC} \end{array} \right\}.$$

Then, we say that ShPIR is (Π, C, ϵ)-secure if for every $n \in \mathbb{N}$ and all $C^* \geq C(n)$, and $I, I' \in [n]^{C^*}$, it holds that:

$$\mathsf{SD}(\widetilde{\mathcal{D}}_{n,\Pi,C^*}(I), \widetilde{\mathcal{D}}_{n,\Pi,C^*}(I')) \leq \epsilon(n).$$

Remark 1 (Randomized number of sub-queries). While Definition 5.1 considers a fixed number of sub-queries k, an interesting consequence of using the shuffle model is that it can support a variable k that is a randomized function of n. In this work, we will only use a fixed k in our main constructions.

Remark 2 (Number of queries v.s. number of clients). We allow clients to make multiple queries; since the queries are anonymous, the server cannot tell whether they are from the same client, and hence the security of PIR in the shuffle model actually relies only on the *total number of queries*, rather than the number of clients. Also, if we require some lower bound on the number of queries, we can let the clients (a given number of them) simply add more dummy queries for arbitrary indices to reach the bound. In formal statements we will always refer to the total number of queries, but for ease of presentation, we may often implicitly assume that there are C clients that each make a single query.

Remark 3 (Adversarial clients). Our model also tolerates adversarial clients who collude with the server. As an extreme example, it is easy to see that if $C - 1$ clients collude with the server, then we are essentially back in the standard single-client setting.
Looking ahead though, our constructions will require a minimum number of *honest* client queries for security. If this is met, security is not reduced by any additional adversarial clients—even an unbounded number of them. Therefore, for simplicity, we can ignore these extra adversarial clients within our analysis.

Efficiency metrics. We measure the efficiency of PIR constructions in the shuffle model using a few metrics below. Since we consider many clients querying the server, we will characterize the cost per query.

- *Per-query (server) computation*: for answering each query, the number of bits that the server reads from the database and the preprocessing bits.
- *Per-query communication*: the sizes of the client query and the server response.
- *Server storage*: the total number of bits, including the preprocessing bits, that are stored by the server.
- *Message complexity*: for each query, the number of anonymous messages required to send. This is separately considered from the communication cost, since we need to take into account the anonymity cost. In particular, this will help us delineate between, e.g., sending one anonymous message of size s and sending s anonymous messages each of size 1 (since the latter may have more network overhead).

While our main focus is the server and the anonymity cost, we may also consider *per-query client computation*, which is the computational complexity for issuing each query and reconstructing the answer. One may also consider *client storage* which is omitted in this work as the clients in our constructions are stateless.

5.1 Warm-up Impossibility Results

When considering PIR with multiple clients, it is useful to study the minimum number of clients required for security. After all, if we have a single client, then under the statistical security notion, this effectively means the client has $\Theta(n)$ communication. We start by showing that for any *linear* PIR , which includes the constructions mentioned in Section 4.1 and others [BIK05, CGKS95], the number of clients required is at least the database size.

We say that a PIR is *linear* if its encoding function is linear; that is, for any two databases x and x', $P_{x+x'} = P_x + P'_x$. Most multi-server PIR schemes (essentially linear smooth locally decodable codes) considered in existing literature are linear.

Theorem 5.2 (Attacks for linear PIR)**.** *Any linear PIR in the shuffle model, when the total number of queries C is less than the database size n, has statistical security no better than $\frac{n-C}{n-1}$.*

At a high level, the attacker simply checks whether or not the value at a given index is determined by the linear constraints imposed by the observed queries. Upon choosing a suitable basis for the domain and range, a linear encoding function can be represented by a generating matrix, which we denote as $\mathbf{M}$. Let M_q denote the row vector corresponding to query q. We will show that when the number of queries is less than n, we can narrow down the set of possible client queries by at least 1.

Lemma 5.3. *Let $e_i \in \mathbb{F}^n$ denote the i-th standard basis vector. Let Q_i^k denote the support of* $\mathsf{Query}(i; n)$*. For every $(q_1, \ldots, q_k) \in Q_i^k$, we have $e_i \in \operatorname{span}\{M_{q_1}, \ldots M_{q_k}\}$.*

Proof of Lemma 5.3. Assume that e_i is not in the span corresponding to $\mathbf{q} = (q_1, \ldots, q_k) \in Q_i^k$. Intuitively, this should mean that the i-th entry of the database cannot be fully determined by these queries. We formalize this intuition in showing that there must exist a database on which Recon fails.

First, we set some notation. Let V denote the vector space spanned by $M_{q_1}, \ldots M_{q_k}$. Let $\mathbf{M_q}$ be the matrix formed by the subrows of $\mathbf{M}$ corresponding to the queries $\mathbf{q}$.

Now we show that there must exist a vector w in the null space of $\mathbf{M_q}$ such that $w_i \neq 0$. We can prove this by contradiction. Assume that for every $w \in \text{null}(\mathbf{M_q})$, $w_i = 0$. In other words, $\text{null}(\mathbf{M_q})$ is orthogonal to e_i. Let $\mathbf{M_q}'$ be $\mathbf{M_q}$ with e_i as an additional row. The previous observation ensures that $\text{null}(\mathbf{M_q}) = \text{null}(\mathbf{M_q}')$. By rank nullity, $\text{rank}(\mathbf{M_q}) = \text{rank}(\mathbf{M_q}')$. We conclude that e_i does not add to the rank of $\mathbf{M_q}$; therefore, e_i can be written as a linear combination of $M_{q_1}, \ldots, M_{q_k}$. This contradicts our assumption that e_i is not in the span of these vectors.

Finally, we show that Recon will fail with some positive probability. Let $x \in \mathbb{F}^n$ be some arbitrary database. Let w be a vector in the null space of $\mathbf{M_q}$ such that $w_i \neq 0$. Then $x_i \neq (x+w)_i$. When given the answers to these particular queries $q_1, \ldots, q_k$, Recon cannot distinguish between x and $x + w$, yet they have different values at index i. Therefore, Recon must fail for at least one of x or $x + w$. □

Proof of Theorem 5.2. Applying the above lemma when the total number of queries is $C < n$, we know that the span of the corresponding row vectors of M will be of dimension at most C, so there will be at least $n - C$ standard basis vectors $e_{i_1}, \ldots e_{i_{n-C}}$ missing from the span. $i_1, \ldots, i_{n-C}$ must not have been in the original set of client indices.

This leads to a natural candidate for a distinguisher. We will show an adversary which has advantage at least $\frac{n-C}{n-1}$ in distinguishing between all-0 vector $(0, \ldots, 0)$ and all-i vector $(i, \ldots, i)$, when the total number of queries is $C < n$. i is some particular index which will depend on the specific PIR.

The distinguisher. If $e_i \notin \mathrm{span}(M_{q_1}, \ldots, M_{q_k})$, then output 0; else output 1.

Claim. *There exists $i \in [n]$ such that the above distinguisher has advantage $\frac{1}{n-1}$ between the all-0 vector and the all-i vector.*

To see why this claim holds, notice that there are the following two cases:

- Case 0 (all-0 vector): in each realization of the queries, there is at least $n - C$ indices whose basis vectors are not in the span of the queries. Since there are $n-1$ of these i's, by linearity of expectation, there must exist some i with probability at least $\frac{n-C}{n-1}$ of being excluded from the span. For the said i, the probability the distinguisher outputs 0 is $\frac{n-C}{n-1}$.
- Case 1 (all-i vector): by Lemma 5.3, e_i is in the span of each of the sharings of i. Therefore, it is in the span of the aggregate of all the shares. The distinguisher outputs 0 with probability 0.

We conclude that the difference in probabilities of the two cases is $\frac{n-C}{n-1}$. □

5.2 Strawman Protocols

Before presenting our main constructions, we describe a few strawman designs.

Split and mix for PIR with additive sub-queries. In the split-and-mix technique [IKOS06], each client query is split into multiple shares; the shares from all queries are then mixed (i.e., shuffled) together before being sent to the server. This can be directly applied to existing additive PIR constructions (e.g., the two-server construction described in Section 4.1.1), i.e., each query will be split into $k \geq 2$ additive shares (or sub-queries). The set of shuffled shares only leaks the sum of the queried encodings but nothing else. Note that it is easy to hide this sum by simply adding one more random share as "noise".

Intuitively, security increases when k becomes larger. However, this directly impacts the communication complexity of the shuffle PIR. Formally, if the encoding $\mathbf{z}$ of a query index is split into k shares, following the notation from Section 4.1.1, the server has to express the encoding polynomial P_x as a km-variate polynomial P'_x and evaluate each share $\mathbf{z}_\ell$ using P'_x. Now, within P'_x, each monomial will have degree at most $(k-1)d/k$ after evaluation which results the protocol having communication cost $O(n^{(k-1)/k})$ which worsens with more additive shares.

The analysis in recent works [GMPV20, BBGN20], although targeted at summation in the shuffle model, can also be adapted for the PIR setting. Both works show that a constant number of additive shares is sufficient to provide security that is some inverse polynomial in the number of clients. Concretely though, their analysis requires each query to be split into at least 4 shares, which would result in high communication cost $O(n^{3/4})$. The question therefore remains if $O(n^{1/2})$ communication can be achieved by using $k = 2$ shares (which will be optimal for the additive sharing technique).

Unfortunately, using only 2 additive shares is not enough for security, as we show below. Recall that the query indices are encoded in the space $\mathbb{F}_2^m$ for some m. Let $i \in [n]$ be the index such that its encoding is 0^m and let $i' \in [n]$ be the index such that its encoding is 1^m. Now consider two sets of query indices, $I = (i, \ldots, i)$ and $I' = (i', \ldots, i')$. Note that a 1 can only be split as $0 + 1$, but a 0 can be split to either as $0 + 0$ or $1 + 1$. Consider first the extreme case where $m = 1$; here, for I', there will always be exactly the same number of 0s and 1s while this is not true for I. A similar issue also exists for a general m; if one share is $\beta \in \mathbb{F}_2^m$, then for I', there will be a corresponding share $1^m - \beta$ (for instance if one share is $(0, 1, 0, 1)$, the other share will be $(1, 0, 1, 0)$). This means that the total number of shares that are β will always be equal to the total number of shares that are $1^m - \beta$. The same will not hold for I, which allows the server to distinguish between I and I'.

Adding noise. A useful observation is that the protocol can have clients add random sub-queries (independent random values, and we call them noise) which can help reduce the statistical distance between the two sets of sub-queries. While this is not possible in the secure summation setting since adding noise will change the sum, and therefore could not be taken advantage of by prior work [GMPV20, BBGN20], doing so compiles with the PIR setting since answers to noise sub-queries can simply be discarded later by clients. A crucial efficiency constraint, however. is that we do not want the total noise to be $\omega(C)$ since this would make the amount of *noise per client* dependent on the total number of clients C.

For the 2-additive PIR, adding a constant amount of noise per client only reduces the statistical distance by a constant factor, i.e., the statistical error does not vanish in C (see Appendix A). In contrast, our solution requires adding only one noise per client, by using a randomization technique, as we will show in Section 6.

6 General Constructions for Single-Server Shuffle PIR

We now present generic ways to build asymptotically efficient PIR protocols in the shuffle model from the PIR constructions mentioned in Section 4.1. The high-level idea is to compose together a protocol OPIR at the *outer* layer with a protocol IPIR at the *inner* layer, for randomizing the query indices. We call this *inner-outer* paradigm for constructing ShPIR protocols.

The insight of having OPIR. Recall from Section 5.2 that security cannot be achieved by just using one PIR with two additive shares; the core problem is that it is easy to distinguish between I and I' that are far apart, e.g., all clients query for index i vs all clients query for index i'. If all client queries were independently random, then this problem would be immediately solved. Our problem is complicated by the fact that the client queries may be arbitrarily correlated, as seen in Section 5.2.

The key insight we use to navigate around this is to first *randomize* the query indices by using a separate *outer* PIR, which we denote as OPIR. The goal of this OPIR protocol is two fold: first, it reduces the distance between the two multi-sets I and I'; and second, it transforms the queries in a way that makes them pairwise-independent which turns out to be sufficient for us to prove security. Concretely, the OPIR protocol takes two multi-sets I and I', who may differ by as much as $\delta = C$, and constructs two new (larger) query multi-sets J and J', whose difference is now proportional to $\sqrt{\delta}$, and whose elements are now pairwise-independent. Then J and J' will be used as query indices of an *inner* PIR with additive (or CNF) shares. In this way, the server sees the IPIR sub-queries as if they were generated from random (and pairwise independent) query indices.

ShPIR **compilation.** To compile the overall ShPIR protocol, the server will need encode the database x twice: once using OPIR and once using IPIR. More precisely, the server first sets up a database consisting of the answers to every possible OPIR sub-queries based on x: it defines a new database $x' = (x'_1, \ldots, x'_{n'})$ of size $n' = |\mathcal{Q}_{\mathsf{OPIR}}|$ where each entry x'_i is set to be $\mathsf{OPIR.Answer}(P_x, L_i)$ where L_i denotes the i-th element in the sorting of $\mathcal{Q}_{\mathsf{OPIR}}$. If $\mathsf{OPIR.Answer}$ is different for different servers, then a size kn' (where k is the number of OPIR servers) database can be used, which concatenates all the n'-sized databases where the ℓ-th database is defined using $\mathsf{OPIR.Answer}^{\ell}(P_x, L_i)$; see Construction 6.1 for details. Now x', from the perspective of IPIR, is the database to be taken into the setup algorithm, i.e., the server runs $\mathsf{IPIR.Setup}(x')$, and the setup for ShPIR is done.

To query an index $i \in [n]$, a client will first use OPIR to generate queries $q_1, \ldots, q_k$ which are each uniformly random in the space $\mathcal{Q}_{\mathsf{OPIR}}$. Each of these q_ℓ can now be treated as an index i'_ℓ of the database x', following which the client will use $\mathsf{IPIR.Query}$ to fetch the i'_ℓ-th entry in x' that corresponds to q_ℓ. As a result, the final sub-queries to be sent to the server (along with additional noise) are generated by the client running $\mathsf{IPIR.Query}$ on the indices i'_ℓ for $\ell \in [k]$. The full details of the composition are given as Construction 6.1.

Section structure. The rest of this section is structured as follows: first, in Section 6.1, we describe a generic construction $\mathsf{Add\text{-}ShPIR}$ that composes together any k-server OPIR with a two-server additive IPIR. Later, in Section 6.2, we show how to reduce the communication complexity by generalizing the IPIR to be based on CNF-sharing; we denote this construction by s-$\mathsf{CNF\text{-}ShPIR}$ when an s-CNF sharing is used. Finally, in Section 6.3, we concretely instantiate these designs using a Reed-Muller code based OPIR.

6.1 Composition with an Additive Two-Server IPIR

We start with our generic composition which uses an IPIR with two additive shares. Our main security result for this composition is given as Theorem 6.1. We provide an overview of the core proof techniques in Section 6.1.1; the full proof is given in Appendix C.

Theorem 6.1 (ShPIR Composition Theorem for additive IPIR)**.** *Let Φ be any k-server t-private information-theoretic PIR scheme where $k > t > 2$; denote its sub-query space size by Q and its answer size by A. Let Ψ be 2-additive PIR defined in Construction 4.1. Then, for any database size $n \in \mathbb{N}$, given any $\epsilon > 0$, there exists a constant c_0 such that for $C \geq (c_0 Q^5)/(k\epsilon^8)$, the construction $\mathsf{ShPIR}(\Phi, \Psi)$ is a (Π, C, ϵ)-secure PIR in the shuffle model where Π is uniform. Here, Q, k, ϵ, C may all be functions of n. Furthermore, when $Q = \widetilde{O}(n)$ and assuming one-time preprocessing, the construction has:*

- *per-query server computation $O(A \cdot k^{\frac{3}{2}} \cdot Q^{\frac{1}{2}})$,*
- *per-query client computation $O(A \cdot k \cdot Q^{\frac{1}{2}})$,*
- *per-query communication $O(A \cdot k^{\frac{3}{2}} \cdot Q^{\frac{1}{2}})$,*
- *server storage $\widetilde{O}(A \cdot k^{\frac{3}{2}} \cdot Q^{\frac{3}{2}})$.*

Remark 4. In Theorem 6.1, we use C to implicitly mean the total number of queries from all *uncorrupted* clients. Furthermore, for any n-bit database, when ϵ is $1/p_1(n)$ for some polynomial p_1, C can be chosen as a polynomial $p_2(n) = c_0(Q(n))^5(p_1(n))^8/k(n)$ for some constant c_0. If C is exponentially large, we can get exponential security.

ShPIR **Composition.** A shuffle model PIR protocol $\mathsf{ShPIR}(\mathsf{OPIR}, \mathsf{IPIR})$ built using the inner-outer paradigm from a k-server OPIR, and a s-server IPIR is defined as follows:

- $\mathsf{ShPIR.Setup}(x) \to P$:
  1. Let $P_x \leftarrow \mathsf{OPIR.Setup}(x)$.
  2. Define a database x' of size n' as follows:
     - Let $n^* = |\mathcal{Q}_{\mathsf{OPIR}}|$ and let $L = (L_1, \ldots, L_{n^*})$ denote the sorting of the sub-query space $\mathcal{Q}_{\mathsf{OPIR}}$.
     - If the Answer algorithm is the same for all OPIR servers:
       For all $i \in [n^*]$, let $x'_i \leftarrow \mathsf{OPIR.Answer}(P_x, L_i)$.
       As a result, x' is of size $n' = n^*$.
     - If the Answer algorithm is different for the k OPIR servers:
       For $i \in [n^*], \ell \in [k]$: let $x'_{i+n' \cdot (\ell-1)} \leftarrow \mathsf{OPIR.Answer}^{\ell}(P_x, L_i)$.
       As a result, x' is of size $n' = kn^*$.
  3. Run $\mathsf{IPIR.Setup}(x')$ and output its result as P.
- $\mathsf{ShPIR.Query}(i; n) \to (q_1, \ldots, q_h)$, where $i \in [n]$ and $h = k(s+1)$:
  1. Initialize $(u_{\ell,j})_{\ell \in [k], j \in [s]}$.
  2. Let $(q'_1, \ldots, q'_k) \leftarrow_{\$} \mathsf{OPIR.Query}(i; n)$.
  3. For $\ell \in [k]$,
     - If the Answer algorithm is the same for all k OPIR servers:
       Map q'_ℓ to the corresponding index $i'_\ell \in [n']$,
       i.e., $x_{i'_\ell} = \mathsf{OPIR.Answer}(P_x, q'_\ell)$.
     - If the Answer algorithm is different for the k OPIR servers:
       Map q'_ℓ to the corresponding index $i'_\ell \in [kn']$,
       i.e., $x_{i'_\ell} = \mathsf{OPIR.Answer}^{\ell}(P_x, q'_\ell)$.
     - Let $(\widetilde{q}_1, \ldots, \widetilde{q}_s) \leftarrow_{\$} \mathsf{IPIR.Query}(i'_\ell; n')$.
     - Set $(u_{\ell,1}, \ldots, u_{\ell,s}) \leftarrow (\widetilde{q}_1, \ldots, \widetilde{q}_s)$.
  4. Let $(r_1, \ldots, r_k) \overset{\$}{\leftarrow} \mathcal{Q}_{\mathsf{OPIR}}$. // dummies
  5. Output $(u_{1,1}, \ldots, u_{k,s}, r_1, \ldots, r_k)$.
- $\mathsf{ShPIR.Answer}(P, q) \to a$:
  1. If IPIR has the same Answer algorithms for server, return $a = \mathsf{IPIR.Answer}(P, q)$; otherwise return
  $$a = \left\{(\mathsf{IPIR.Answer}^{\ell}(P, q), \mathsf{label}\ \ell)\right\}_{\ell \in [s]}.$$
- $\mathsf{ShPIR.Recon}(a_1, \ldots, a_h) \to x_i$:
  1. Initialize $(v_{\ell,j})_{\ell \in [k], j \in [s]}$ and $(a'_\ell)_{\ell \in [k]}$.
  2. For $\ell \in [k]$, $j \in [s]$:
     - Let $a_{(\ell-1) \cdot k + j}$ be the answer to sub-query $q_{(\ell-1) \cdot k + j}$, namely $u_{\ell,j}$.
     - If IPIR has different Answer algorithms for the servers, parse $a_{(\ell-1) \cdot k + j}$ as
     $$\{(\widetilde{a}_1, \mathsf{label}\ 1), \ldots, (\widetilde{a}_s, \mathsf{label}\ s)\}, \text{ let } v_{\ell,j} := \widetilde{a}_j \text{ (whose associated label is } j).$$
     - If IPIR has the same Answer algorithms for the servers, let $v_{\ell,j} = a_{(\ell-1) \cdot k + j}$.
  3. For $\ell \in [k]$:
     - $a'_\ell \leftarrow \mathsf{IPIR.Recon}(v_{\ell,1}, \ldots, v_{\ell,s})$.
  4. Output $x_i \leftarrow \mathsf{OPIR.Recon}(a'_1, \ldots, a'_k)$.

Construction 6.1: Composed ShPIR built using the inner-outer paradigm.

Remark 5 (Answering IPIR sub-queries). Recall that in the 2-additive PIR protocol (Figure 4.1), the servers respond to a client's sub-query knowing which server it acts for: after encoding the database as a $2m$-variate polynomial P'_x containing a set $\mathcal{M}$ of monomials, the first server evaluates only those monomials from a fixed set $\mathcal{M}_1$ at the client sub-query, while the second server evaluates monomials from the set $\mathcal{M}_2 = \mathcal{M} \setminus \mathcal{M}_1$. This means that it must be known to the servers whether they are the "first" or "second" server in the protocol. Consequently, when compiling this as the inner layer of our ShPIR construction, since there is only one server, it needs to figure out which shares to evaluate using $\mathcal{M}_1$ and which using $\mathcal{M}_2$.

One idea is to have the client label the shares; this significantly complicates the analysis since there is now additional structure. Instead, we have the server answer each share twice: once according to $\mathcal{M}_1$, and once according to $\mathcal{M}_2$ and send back the tuple as its response. Since the client knows which was the first share and which was the second, it can pick the correct responses to be used for reconstruction. This only results in a $2\times$ blowup in the server communication. This is formally showed in Figure 6.1.

Remark 6 (Reduced cost for homogeneous servers). For similar reason as above, if OPIR has different Answer algorithms for the servers, the ShPIR server needs to store k sub-databases, where for ℓ-th sub-database the server treats $q \in \mathcal{Q}_{\mathsf{OPIR}}$ as the ℓ-th share and stores the corresponding answers. If $\mathsf{OPIR.Answer}$ is the same for all k servers, then ShPIR server only needs to store one such sub-database; as a result, both the per-query server computation and communication will be $O(A \cdot k \cdot Q^{\frac{1}{2}})$, and the server storage will be $O(A \cdot Q^{\frac{3}{2}})$. The client computation will be $O(A \cdot k \cdot Q^{\frac{1}{2}})$. See details in Appendix C.4.

6.1.1 Proof Outline of Theorem 6.1

Basic background. Consider a client query index $i \in [n]$. Recall that our k-server OPIR will first encode i into the space $\mathcal{E}_{\mathsf{OPIR}}$ and then split it into k sub-queries in the space $\mathcal{Q}_{\mathsf{OPIR}}$. When composing with the IPIR, these k sub-queries will now be interpreted as IPIR query indices within the IPIR database of size $|\mathcal{Q}_{\mathsf{OPIR}}|$. Each of the k indices will now be encoded within the IPIR encoding space $\mathcal{E}_{\mathsf{IPIR}}$, and then split into 2 shares in the space $\mathcal{Q}_{\mathsf{IPIR}}$. Note that the space $\mathcal{Q}_{\mathsf{OPIR}}$ and $\mathcal{E}_{\mathsf{IPIR}}$ have the same size, which is the size of the IPIR database, and that $\mathcal{E}_{\mathsf{IPIR}} \subset \mathcal{Q}_{\mathsf{IPIR}}$. Going forward, for clarity, we keep using "sub-queries" for OPIR but use "shares" to mean the sub-queries for IPIR.

Given C clients, we will have kC total IPIR query indices encoded into $\mathcal{E}_{\mathsf{IPIR}}$; denote this by $\mathsf{y} = (y_1, \ldots, y_{kC})$ and let $\widetilde{\mathsf{y}}$ (of length $2kC$) denote its shares in $\mathcal{Q}_{\mathsf{IPIR}}$. Our main goal is to analyze the properties of $\widetilde{\mathsf{y}}$ since this will be the view of the server. In particular, given two lists of original query indices $I = (i_1, \ldots, i_C)$ and $I' = (i'_1, \ldots, i'_C)$, and their resulting shares $\widetilde{\mathsf{y}}$ and $\widetilde{\mathsf{y}}'$, we want to understand whether an adversary can find e.g., which of I or I' corresponds to $\widetilde{\mathsf{y}}$.

Balls-and-bins-formulation. We now describe how to formulate our core analysis as a balls-and-bins problem. A key starting observation here is that a uniformly random shuffler Π will eliminate any ordering within $\widetilde{\mathsf{y}}$ (and similarly for y). In turn, this allows us to essentially do our analysis using a balls-and-bins formulation, where each share in $\widetilde{\mathsf{y}}$ corresponds to a ball in one of $|\mathcal{Q}_{\mathsf{IPIR}}|$ bins. More precisely, the distribution of the shuffled shares in $\widetilde{\mathsf{y}}$ is exactly a $|\mathcal{Q}_{\mathsf{IPIR}}|$-dimensional distribution where the each component represents the distribution of the number of balls in that bin. Towards this, we also find it helpful to analyze y using a similar balls-and-bins formulation.

The crux of our analysis now boils down to quantifying the statistical distance between the distribution of balls over bins resultant from any two sets of original query indices I and I'. Specifically, define $\mathcal{Y}(I)$ to be the distribution of the balls-and-bins configuration of IPIR query indices y resultant from the original query indices I; define $\widetilde{\mathcal{Y}}(I)$ to be the distribution of its shares (i.e., corresponding to $\widetilde{\mathsf{y}}$). Roughly, the goal now is to show that for any I and I', we can bound $\mathsf{SD}(\widetilde{\mathcal{Y}}(I), \widetilde{\mathcal{Y}}(I'))$ with some inverse polynomial in the number of clients.

Looking ahead however, for our proof to go through, we will require some extra balls to be added uniformly at random, essentially to "smooth out" the distribution of $\widetilde{\mathsf{y}}$; this can also be thought of as uniformly random *noise*. In the PIR context, this effectively corresponds to each client sending a random sub-query in $\mathcal{Q}_{\mathsf{IPIR}}$. We denote the balls-and-bins distribution of the shares with noise added as $\widetilde{\mathcal{Y}}^*(I)$.

Remark 7 (Noise and communication complexity). We note that adding noise for each IPIR query index does not increase the asymptotic communication complexity for IPIR, i.e., the communication for an n-sized database is still $O(\sqrt{n})$. This is because the server will still evaluate each noise share either as the first or second share without changing the database encoding polynomial making the communication still $O(\sqrt{n})$. Note that adding noise is substantially different from splitting to more shares, i.e., if each IPIR index was instead split into more additive shares (corresponding to using an IPIR with more servers), then the number of variables in the encoding polynomial itself will be larger, which would increase the asymptotic communication.

Main proof steps. At a high level, we leverage balls-and-bins style analyses to bound the statistical distance between $\widetilde{\mathcal{Y}}^*(I)$ and $\widetilde{\mathcal{Y}}^*(I')$. The rough idea will be to first compute the *edit distance* between the balls-and-bins configurations corresponding to the IPIR shares and then use that to bound the statistical distance after adding the random noise. Our proof proceeds in three major steps which we outline below.

Proof Step 1: *(Analyzing the edit distance of* OPIR *sub-queries; Appendix C.1).* Consider two lists of client indices $I = (i_1, \ldots, i_C)$ and $I' = (i'_1, \ldots, i'_C)$. Abstractly, the first part of our proof shows that the edit distance between the OPIR sub-queries generated from I and I' is *not too large.*

Recall that the t-out-of-k OPIR sub-queries generated are individually uniformly random, and are $(t-1)$-wise independent (and therefore also pairwise independent). Therefore, we can formulate our objective as the following balls-and-bins problem given in Lemma 6.2.

Lemma 6.2. *Suppose that B balls are thrown into N bins. Let $\mathcal{B}$ and $\mathcal{B}'$ be any two distributions of the final balls-and-bins configuration where each ball is thrown uniformly at random, and any two balls are independently thrown. Then:*

$$\mathbb{E}_{\mathbf{u}\sim\mathcal{B},\mathbf{v}\sim\mathcal{B}'}[\mathsf{ED}(\mathbf{u}, \mathbf{v})] \leq \sqrt{\frac{BN}{2}}.$$

Casting this result to our construction, since each client index generates k OPIR sub-queries and there are C clients in total, the expectation of edit distance (or differences) between any two sets of OPIR sub-queries (and consequently, the IPIR indices) is at most $\sqrt{kC\left|\mathcal{Q}_{\mathsf{OPIR}}\right|/2}$.

Proof Step 2: *(Analyzing the edit distance of 2-additive sharing in the* IPIR*; Appendix C.2).* Now that we have a bound on the edit distance between OPIR sub-queries (and consequently IPIR

indices), our next step is to analyze the edit distance for shares in $\mathcal{Q}_{\mathsf{IPIR}}$. Recall that each encoded index in $\mathcal{E}_{\mathsf{IPIR}}$ is split into two additive shares. We model this as another balls-and-bins problem below.

Consider a (B, N)-valid configuration $\mathbf{u}$ and let $\mathsf{Share}_{\mathbf{u}}$ denote the distribution of randomly splitting each ball in $\mathbf{u}$ (in a group $\mathbb{G}$), i.e., for each ball b, throw one ball into a random bin $u \leftarrow\$\ \mathbb{G}$, and another into bin $b - u$. The goal now is to bound the edit distance between $\mathsf{Share}_{\mathbf{u}}$ and $\mathsf{Share}_{\mathbf{v}}$ given the edit distance between $\mathbf{u}$ and $\mathbf{v}$.

To begin, we first show that in the context of the final statistical distance, it is sufficient to only consider the parts of $\mathbf{u}$ and $\mathbf{v}$ that are different. Let $\mathsf{Share}^{\ell}_{\mathbf{u}}$ denote the distribution of the balls-and-bins configuration when further throwing ℓ balls independently and uniformly at random following the sharing $\mathsf{Share}_{\mathbf{u}}$. In particular, we show (in Lemma C.1; Appendix C.2) that,

$$\mathsf{SD}(\mathsf{Share}^{\ell}_{\mathbf{u}}, \mathsf{Share}^{\ell}_{\mathbf{v}}) \leq \mathsf{SD}(\mathsf{Share}^{\ell}_{\mathbf{u}\ominus\mathbf{v}}, \mathsf{Share}^{\ell}_{\mathbf{v}\ominus\mathbf{u}})$$

where $\ominus$ denotes the ball-difference operation defined in Section 4.2. Essentially, this will allow us to look at the splitting of only those balls that differ between $\mathbf{u}$ and $\mathbf{v}$; in particular, given (B, N)-valid $\mathbf{u}$ and $\mathbf{v}$ with edit distance δ, we will only need to concern ourselves with the (δ, N)-valid $\mathbf{u}' = \mathbf{u} \ominus \mathbf{v}$ and $\mathbf{v}' = \mathbf{v} \ominus \mathbf{u}$. We show the following result (in Lemma C.2; Appendix C.2):

$$\mathbb{E}\left[\mathsf{ED}(\mathsf{Share}_{\mathbf{u}'}, \mathsf{Share}_{\mathbf{v}'})\right] \leq \sqrt{2\delta N}.$$

Combining this with the result from the first proof part, we get:

$$\begin{aligned}\mathbb{E}_{\mathbf{u}\sim\mathcal{B},\mathbf{v}\sim\mathcal{B}'}\left[\mathsf{ED}(\mathsf{Share}_{\mathbf{u}\ominus\mathbf{v}}, \mathsf{Share}_{\mathbf{v}\ominus\mathbf{u}})\right] &\leq \sqrt{2N} \cdot \mathbb{E}_{\mathbf{u}\sim\mathcal{B},\mathbf{v}\sim\mathcal{B}'}\left[\sqrt{\mathsf{ED}(\mathbf{u}, \mathbf{v})}\right] \\ &\leq \sqrt{2N}\,(BN/2)^{1/4} = (2)^{1/4} B^{1/4} (N)^{3/4}\end{aligned}$$

where the second step is by the concave Jensen's inequality.

Proof Step 3: *(Bounding the final statistical distance).* We are now ready to bound the final statistical distance between the final views of the server: $\widetilde{\mathcal{Y}}^*(I)$ and $\widetilde{\mathcal{Y}}^*(I')$. For this, we leverage a recent analysis by Boyle et al [BGIK22]. A straightforward corollary of their result can be abstractly stated as follows: Consider ℓ balls thrown independently and uniformly at random into N bins and let $\mathcal{U}_j$ denote the final distribution after another ball is added into bin j. Then for all bins j and j', we have $\mathsf{SD}(\mathcal{U}_j, \mathcal{U}_{j'}) \leq \sqrt{N/\ell}$. Informally, this can also be thought of as a "toy in sand" problem of being able to hide the location (bin j or bin j') of an initial ball (i.e., the toy) after throwing in N random balls as noise (i.e., the sand). The same analysis can be extended to show that if there are Δ initial balls, after which ℓ random balls are thrown, the statistical distance will be bounded by $\Delta \cdot \sqrt{N/\ell}$. In the context of our PIR analysis, intuitively, Δ will represent the edit distance between $\mathsf{Share}_{\mathbf{u}\ominus\mathbf{v}}$ and $\mathsf{Share}_{\mathbf{v}\ominus\mathbf{u}}$, while the ℓ extra balls will represent the additional "noise" IPIR queries made. Note that when using this balls-and-bins analysis, we need to account for the fact that the edit distance is a distribution in our case, rather than a fixed number; it is straightforward to do so by using standard first-moment techniques (since we have a bound on the expectation).

Casting these analyses back to our PIR context, first notice that $\widetilde{\mathcal{Y}}^*(I)$ is nothing but the distribution $\mathsf{Share}^{\ell}_{\mathbf{u}\sim\mathcal{B}(I)}$ where $\mathcal{B}(I)$ is the distribution of OPIR sub-queries resulting from the indices I.

Looking ahead, we will use $\ell = kC$ uniformly random IPIR queries (i.e., k per client) as noise. A crucial point here is that the number of extra balls per client needs to be constant in C so that the individual communication complexity of each client does not depend on the how many clients are making queries. In fact, this also required our bound on the ED of the 2-additive sharing to be $o(\delta)$.

Combining the results from the previous parts, we show our main result:

$$\mathsf{SD}(\widetilde{\mathcal{Y}}^*(I), \widetilde{\mathcal{Y}}^*(I')) < \frac{3 \cdot N^{5/8}}{B^{1/8}} = \frac{3\,|\mathcal{Q}_{\mathsf{IPIR}}|^{5/8}}{(kC)^{1/8}}.$$

since $N = |\mathcal{Q}_{\mathsf{IPIR}}|$ bins (query-space) and $B = kC$ balls (total sub-queries).

A final complication is bounding $|\mathcal{Q}_{\mathsf{IPIR}}|$ by Q (i.e., the size of OPIR sub-query space). We defer the details to Appendix E.1 (Lemma E.1); the high-level idea is that we let each IPIR database entry be A bits and consequently $|\mathcal{Q}_{\mathsf{IPIR}}|$ can be made $\widetilde{O}(Q)$. Then, assuming that there are $C = \Omega(n^{5+\nu}/k)$ client queries for some constant $\nu > 0$, the statistical distance can be bounded by some inverse polynomial $1/\mathsf{poly}(n)$ in n. More specifically, suppose that we wanted to bound the statistical distance by some inverse polynomial $\epsilon(n)$. Then, assuming at least $C(n) = \Omega(n^5/(k \cdot \epsilon^8))$ client queries, the statistical distance is bounded by ϵ. Consequently, the construction satisfies (Π, C, ϵ)-security in the shuffle model where Π is the uniform shuffler. This completes the proof.

Remark 8 (Concrete trade-off between the number of clients and the amount of noise). Recall that in the final step for bounding the statistical distance, we added kC balls in total, i.e., k independent random noise sub-queries for each client. We can, in fact, add just one random noise for each client and achieve the same level of security but at the cost of increasing the concrete number of clients required by a factor of k^2.

The reverse can be done as well; by adding more noise per client, say γk, the concrete number of clients can be reduced by a factor of γ^2 at the cost of increasing the communication of each client by a factor of γ (which would be asymptotically identical when γ is a constant). This is expected since intuitively noise sub-queries provide more randomness than arbitrary client queries.

6.2 Reducing Communication using CNF Shares

In this section, we describe how to generalize the IPIR to use CNF shares instead of additive shares. The upshot is that it allows us to reduce the communication complexity of the resultant ShPIR protocol to $O(n^c)$ for any constant $c > 0$.

Construction outline. Previously in Section 4.1.1, when looking at a standard multi-server PIR, we mentioned how using s additive shares instead of 2 results in an increased communication cost of $O(n^{(s-1)/s})$ but this can be reduced to $O(n^{1/s})$ at the cost of a stronger non-collusion assumption using a CNF sharing where each server is given a different $s-1$ sized subset of the additive shares. We show that the same strategy in fact also works in our inner-outer paradigm by using an IPIR with CNF-shares (the composed protocol is given in Figure 4.2). This compilation is particularly interesting since it requires *no extra non-collusion assumptions* to get the gain in efficiency (since the shuffle model already consists only of a single server). Instead, the trade-off will arise in the minimum number of clients required for security.

An s-CNF IPIR can simply be used as a drop-in replacement into Construction 6.1 to obtain a composed shuffle model protocol s-CNF-ShPIR. Here, upon obtaining the OPIR sub-queries, the client splits the encoding $\mathbf{z}$ into s additive shares $\mathbf{z}_1, \ldots, \mathbf{z}_s$ (in a group $\mathbb{G}$), and then constructs s

CNF shares where the i-th share is $(\mathbf{z}_{i+1}, \ldots, \mathbf{z}_s, \mathbf{z}_1, \ldots, \mathbf{z}_{i-1}) \in \mathbb{G}^{s-1}$ (see details in Figure 4.2). The CNF shares, i.e., sub-queries for IPIR, are then sent to the single server in s-CNF-ShPIR.

Theorem 6.3 shows the security and efficiency of this composition. We provide an outline of the proof in Section 6.2.1 and defer the full proof to Appendix D.

Theorem 6.3 (ShPIR Composition Theorem for CNF IPIR)**.** *Let Φ be any k-server t-private information-theoretic PIR scheme where $k > t > 2$; denote its sub-query space size by Q and its answer size by A. Let Ψ be the s-CNF PIR defined in Construction 4.2. Then, for any database size $n \in \mathbb{N}$, and given any $\epsilon > 0$, there exists a constant c_0 such that for $C \geq (c_0 Q^{2s+1})/(k\epsilon^8)$, the construction $\mathsf{ShPIR}(\Phi, \Psi)$ is a (Π, C, ϵ)-secure PIR in the shuffle model where Π is uniform. Here, Q, k, ϵ, C may all be functions of n. Furthermore, when $Q = \widetilde{O}(n)$ and assuming one-time preprocessing, the construction has:*

- *per-query server computation $O(A \cdot k^{1+1/s} \cdot Q^{1/s})$,*
- *per-query client computation $O(A \cdot k \cdot Q^{1/s})$,*
- *per-query communication $O(A \cdot k^{1+1/s} \cdot Q^{1/s})$,*
- *server storage $\widetilde{O}(A \cdot k^{1+1/s} \cdot Q^{1+1/s})$,*

Similar to Remark 6, if OPIR.Answer is the same for all k servers, then both the per-query server computation and communication will be $O(A \cdot k \cdot Q^{1/s})$, and the server storage will be $O(A \cdot Q^{1+1/s})$. The client computation will be $O(A \cdot k \cdot Q^{1/s})$.

6.2.1 Proof Outline of Theorem 6.3

The overall structure of the proof is very similar to that of the composition theorem for additive IPIR; recall that there were three major steps.

We start by observing that since the first step deals solely with OPIR, it will be identical in this proof for s-CNF-ShPIR. Furthermore, the third step is identical as well since it involves bounding the overall SD once we have a bound on the ED after the IPIR sharing. The only part of the proof that needs to change therefore, is the second step which deals with bounding the ED after the IPIR sharing—specifically the CNF-sharing in our case.

Define s-CNF-Share$_{\mathbf{u}}$ to be the distribution of the balls-and-bins configuration upon sharing each ball in $\mathbf{u}$ into s CNF shares in $\mathbb{G}^{s-1}$. Now, given (δ, N)-valid configurations $\mathbf{u}$ and $\mathbf{v}$, we want to bound the edit distance between s-CNF-Share$_{\mathbf{u}}$ and s-CNF-Share$_{\mathbf{v}}$; moreover, similar to the proof for Add-ShPIR, this bound needs to be $o(\delta)$ to ensure that the communication complexity of each client does not depend on the total number of clients. The key challenge here turns out to be understanding the (cyclic rotational) symmetries in the CNF-sharing and how they manifest into the distribution of ball-and-bins configuration after the IPIR sharing.

Cyclic symmetries in the CNF-sharing. Concretely, we want to analyze the following: given a ball in bin b, when this ball is split in s CNF-shares, for any bin $\alpha \in \mathbb{G}^{s-1}$ (corresponding to the new bins after the IPIR sharing):

- What is the probability that one of the CNF-shares falls in bin α? Intuitively, this is proportional to the number of distinct cyclic rotations of α.

- Given that one of the CNF-shares falls in bin α, due to the symmetries of CNF-sharing how many other shares are also forced to fall in bin α? Intuitively, this is the number of cyclic symmetries of α.

It turns out this problem has a natural group theoretic flavor; very abstractly, the group of cyclic rotations is isomorphic to the additive group $\mathbb{Z}_s$, the (sub)-group of cyclic symmetries of α is isomorphic to the subgroup $\langle c \rangle \subset \mathbb{Z}_s$ generated by the smallest cyclic symmetry c, while the number of distinct rotations of α is the number of cosets of $\langle c \rangle$ in $\mathbb{Z}_s$. Such a formulation allows us to directly use Lagrange's theorem; the upshot being a clean representation of the distribution for the number of CNF-shares in any bin α. Concretely, this allows us to show the following result (Lemma D.4):

$$\mathsf{ED}(s\text{-}\mathsf{CNF}\text{-}\mathsf{Share}_{\mathbf{u}}, s\text{-}\mathsf{CNF}\text{-}\mathsf{Share}_{\mathbf{v}}) \leq sN^{(s-1)/2}\sqrt{\delta}.$$

Notice that this bound nicely captures the bound on the edit distance correpsonding to Share in the 2-additive IPIR construction. Once we have this bound, the rest of the security proof of proceeds in exactly the same way as the one for Add-ShPIR. Finally, for any $\epsilon > 0$, there is some constant c_0 such that the protocol achieves ϵ-security when there are at least $C \geq (c_0 Q^{2s+1})/(k\epsilon^8)$ client queries where k denotes the number of OPIR servers.

6.3 Concrete Constructions based on Reed-Muller Code

We use Theorem 6.3 and instantiate OPIR with concrete protocols to derive our main results. To minimize the answer size, we use the k-server protocol with Shamir shares described in Section 4.1 (we interchangeably call it Reed-Muller code) to instantiate OPIR. To reduce the communication, we instantiate IPIR with the s-server CNF PIR protocol in Figure 4.2.

Parameters. We now discuss how to pick parameters for the composed PIR scheme that results in our main theorem. Now let OPIR be the k-server Reed-Muller PIR described in Section 4.1.2 (details in Figure 4.3); and IPIR be the 2-additive PIR (Section 4.1.1, Figure 4.1) or s-CNF PIR (Figure 4.2). Note that the 2-additive PIR is a special case of s-CNF PIR. Below we give a two-step overview of choosing parameters; Appendix E.1 gives a more fine-grained choice.

Let m, d, k, t be parameters for OPIR; recall that m is the number of variables, d is the polynomial degree, k is the number of OPIR servers and t is the privacy threshold. Let s, m', d' be parameters for IPIR, where s is the number of IPIR servers, m' is the number of variables and d' is the polynomial degree.

Step 1: *The* IPIR *database size resulting from* OPIR. Recall that IPIR database size n' depends on the size of sub-query space of OPIR. Now we show how to pick m, d, k, t for OPIR in order to get an $\Theta(n)$-sized IPIR database. We first choose m, d, t; and depending on them, choose $|\mathbb{F}|$ and k. The primary requirement is $\binom{m+d}{d} \geq n$ (see details in Section 4.1). Let m and t both be constants larger than 2, then the degree $d = O(n^{1/m})$. Secondly, we require that $|\mathbb{F}| > k > td$, and there exists k and $|\mathbb{F}|$ that are $O(n^{1/m})$ that makes this requirement holds. Suppose $|\mathbb{F}| = c \cdot n^{1/m}$ for some constant c. The space $\mathcal{Q}_{\mathsf{OPIR}}$ is of size $|\mathbb{F}|^m = c^m \cdot n$, and since c, m are both constant, we have $|\mathcal{Q}_{\mathsf{OPIR}}| = \Theta(n)$. Let the $\mathbb{F}$ also be the field of database elements in IPIR (the field of OPIR), then the IPIR database consists of $n' = \Theta(n)$ entries with each entry of size $|\mathbb{F}| = \Theta(n^{1/m})$.

Step 2: *Preprocessing the* IPIR *database.* As we mentioned in Section 3, the server can pre-compute all the answers and store them as a lookup table. We now want to make the preprocessing for

IPIR possible—we need to ensure the size of sub-query space $\mathcal{Q}_{\mathsf{IPIR}}$ is polynomial in n. First, from above, there exists a constant c' such that the IPIR database x' has size $n' = c' \cdot n$; and according to Section 4.1, there exists constants c_1', c_2' such that choosing $m' = c_1' \log n$ and $d' = c_2' \log n$ can ensure $\sum_{\ell=0}^{d'} \binom{m'}{\ell} \geq n$. Since each sub-query (CNF share) in IPIR is a vector with size $s-1$, therefore the size of $\mathcal{Q}_{\mathsf{IPIR}}$ (i.e., the number of entries in the lookup table) is $2^{m'(s-1)}$, which is $n^{c_1'(s-1)}$. When s is a constant, the server can pre-compute all answers to the sub-queries in polynomial time.

Each answer polynomial in IPIR has number of monomials $O(n^{1/s})$ where the coefficients of the monomials are in $|\mathbb{F}|$. Therefore, the number of bits of each answer is $\widetilde{O}(n^{1/s})$; the total number of preprocessing bits is bounded by $n^{c_1' s}$.

Remark 9. Typically for Reed-Muller PIR (Figure 4.3), we choose the parameters m, d, k, t such that we can achieve polylogarithmic communication complexity with the minimum number of servers. However, in the inner-outer PIR composition, we want to make inner PIR database size $\Theta(n)$ so that we get $O(n^{1/s})$ communication, therefore we choose the number of servers to be $\Theta(n^{1/m})$, which is not as good as the typical case where there is polylogarithmic servers.

Theorem 6.4. *For every constant $0 < \gamma < 1$, there exists a Reed-Muller PIR Φ and a $(\lceil 2/\gamma \rceil)$-CNF PIR Ψ, such that on any database size $n \in \mathbb{N}$, given any $\epsilon > 0$, for all $C \geq c_0 n^{1+4/\gamma}/\epsilon^8$ where c_0 is some constant, the construction $\mathsf{ShPIR}(\Phi, \Psi)$ is a (Π, C, ϵ)-secure PIR where Π is uniform. Furthermore, assuming one-time preprocessing, the construction has*

- *per-query server computation $O(n^\gamma)$,*
- *per-query client computation $O(n^\gamma)$,*
- *per-query communication $O(n^\gamma)$,*
- *per-query message complexity $O(n^\gamma)$,*
- *server storage is $\widetilde{O}(n^{1+\gamma/2})$.*

We defer the full proof of Theorem 6.4 to Appendix E.1. One thing to note here is that the reduced communication per client with CNF shares comes at a price—to achieve the same level of security, we need a larger number of clients.

Remark 10 (Sub-polynomial communication assuming super-polynomial number of clients). An interesting consequence of the CNF-based IPIR is that it also enables more efficient protocols in the shuffle model. Using a $(\log n)$-server CNF-based protocol as our IPIR, we can achieve communication of $O(\mathsf{polylog}(n))$ with the assumption that there are at least some super-polynomial $n^{O(\log n)}$ number of clients. This results in better asymptotic complexity than the best existing protocols [DG15] in the standard-model PIR which use a constant numbers of servers. Note that the shuffle model compilation means that still only one server is required for our protocol and therefore we do not require the non-collusion assumptions of the standard-model CNF-based PIR.

Remark 11 (Negligible security with slightly sublinear communication). Our main result only achieves inverse-polynomial rather than negligible security error. We note that if one settles for *slightly sublinear* communication, there is a simple solution that achieves negligible security error and proceeds as follows. The server writes the n-bit database as an $m \times m$ matrix over $\mathbb{Z}_2$ where $m = \sqrt{n}$. Each client writes the column it is interested in as a unit vector $q \in \mathbb{Z}_2^m$. Assuming C clients query at the same time, where C is super-linear in n, each client splits the vector q into $k = O((m + \sigma)/\log C)$ additive shares, for security parameter $\sigma = \log^2 n$. For each query $q' \in \mathbb{Z}_2^m$, the server responds with $X \cdot q' \in \mathbb{Z}_2^m$. By the tight security analysis of the *additive* split-and-mix protocol [IKOS06, BBGN20, GMPV20], the security error is negligible in n, i.e., $\Theta(1/n^{\log n})$, and both the query and the answer are of size $k \cdot m = O(n/\log n)$.

6.4 Combining with Standard-Model PIR

While this work focuses on theoretical feasibility results, it can be used as a blackbox to reduce server cost for standard single-server PIR protocols by any constant factor which may result in significant savings in practice (even $10\times$ is a concretely substantial improvement). The result will still be a PIR protocol in shuffle model, but with reduced number of clients compared to using shuffle PIR alone and reduced server cost compared to using standard single-server PIR alone. We give the idea below.

Take any standard single-server PIR scheme stdPIR and denote the shuffle PIR construction as ShPIR. The server organizes the size-n database as an $\ell \times (n/\ell)$ matrix where ℓ is a constant. The key idea here is to use stdPIR to retrieve a column and ShPIR to retrieve a row. The server treats each column as a database in ShPIR and runs $\mathsf{ShPIR.Setup}$ on it. The server stores the preprocessed results as lookup tables (hence n/ℓ tables in total).

Suppose a client wants to retrieve the entry at r-th row and c-th column. The client runs the query algorithm of ShPIR on index $r \in [\ell]$ and generates k sub-queries. Then the client sends k messages anonymously, where the j-th message consists of the j-th sub-query of ShPIR and a stdPIR query for index $c \in [n/\ell]$. On receiving each message, the server first processes the sub-query of ShPIR (essentially n/ℓ table lookup operations), which results in n/ℓ elements; then the server processes the stdPIR query on these n/ℓ elements.

Compared to running stdPIR on a size-n database, this technique reduces server computation by a factor of ℓ. And the ShPIR database size is ℓ, which neither requires too many clients nor incurs high anonymity cost. The tradeoff is that a client sends k messages in the stdPIR-ShPIR combination instead of one message when using stdPIR only.

6.5 Lower Bound on Security

We show that for shuffle PIR protocols constructed in the inner-outer paradigm, $1/\mathsf{poly}(n)$ statistical security is tight in the sense that negligible security cannot be achieved with polynomially many clients using the additive inner PIR. The proof is deferred to Appendix F.

This result does not rule out the information-theoretic constructions with negligible error, in particular, an interesting open problem to consider is instantiating the inner PIR with the Reed-Muller construction.

Theorem 6.5 (Lower bound on security for ShPIR)**.** *Let Φ be any multi-server PIR scheme. Denote the number of possible vectors of sub-queries as K_Φ. Let Ψ be a constant-server additive PIR (Construction 4.1). On any database size $n \in \mathbb{N}$, for all (Π, C, ϵ)-secure $\mathsf{ShPIR}(\Phi, \Psi)$ constructions where C, K_Φ and K_Ψ are all bounded by polynomial $p_1(n)$, there exists a polynomial p_2 such that $\epsilon \geq 1/p_2(n)$.*

7 Conclusion and Open Questions

We demonstrate that PIR in the shuffle model can circumvent several limitations of standard-model PIR. This includes information-theoretic security with a single server, which opens a direction of constructing concretely efficient single-server schemes in the future.

The main technical question we leave open in this work is the possibility of obtaining similar results with negligible security error (recall that we can achieve this with slightly sublinear communication, see Remark 11). We conjecture that polylogarithmic communication per client with negligible security can be achieved by instantiating both OPIR and IPIR with the Reed-Muller PIR construction with a polylogarithmic security threshold and a polylogarithmic communication complexity.

Finally, an interesting direction for future research is obtaining concretely efficient PIR schemes in the shuffle model, possibly by settling for computational security. A first step of this direction was recently taken in [GIK+24].

Acknowledgements. This research was supported by a Google faculty grant. Yuval Ishai was additionally supported by ERC Project NTSC (742754), BSF grants 2018393 and 2022370, ISF grant 2774/20, and ISF-NSFC grant 3127/23. Mahimna Kelkar and Yiping Ma were partially supported by a Technion research scholarship. Yiping Ma was also supported by a Microsoft Research PhD Fellowship.

References

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

[AIK+21] Shweta Agrawal, Yuval Ishai, Eyal Kushilevitz, Varun Narayanan, Manoj Prabhakaran, Vinod M. Prabhakaran, and Alon Rosen. Secure computation from one-way noisy communication, or: Anti-correlation via anti-concentration. In *Proceedings of the International Cryptology Conference (CRYPTO)*, pages 124–154, 2021.

[AIVG22] Kinan Dak Albab, Rawane Issa, Mayank Varia, and Kalman Graffi. Batched Differentially Private Information Retrieval. In *Proceedings of the USENIX Security Symposium*, pages 3327–3344, 2022.

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

[AMBFK16] Carlos Aguilar-Melchor, Joris Barrier, Laurent Fousse, and Marc-Olivier Killijian. XPIR: Private Information Retrieval for Everyone. In *Proceedings of the Privacy Enhancing Technologies Symposium (PETS)*, 2016.

[APY20] Ittai Abraham, Benny Pinkas, and Avishay Yanai. Blinder - scalable, robust anonymous committed broadcast. In *Proceedings of the ACM Conference on Computer and Communications Security (CCS)*, pages 1233–1252, 2020.

[BBG23] Borja Balle, James Bell, and Adrià Gascón. Amplification by shuffling without shuffling, 2023. `https://arxiv.org/pdf/2305.10867.pdf`.

[BBGN20] Borja Balle, James Bell, Adria Gascon, and Kobbi Nissim. Private summation in the multi-message shuffle model. In *Proceedings of the ACM Conference on Computer and Communications Security (CCS)*, 2020.

[BEM$^+$17] Andrea Bittau, Ulfar Erlingsson, Petros Maniatis, Ilya Mironov, Ananth Raghunathan, David Lie, Mitch Rudominer, Ushasree Kode, Julien Tinnes, and Bernhard Seefeld. PROCHLO: Strong Privacy for Analytics in the Crowd. In *Proceedings of the ACM Symposium on Operating Systems Principles (SOSP)*, 2017.

[BGIK22] Elette Boyle, Niv Gilboa, Yuval Ishai, and Victor I. Kolobov. Programmable Distributed Point Functions. In *Proceedings of the International Cryptology Conference (CRYPTO)*, 2022.

[BIK05] Amos Beimel, Yuval Ishai, and Eyal Kushilevitz. General constructions for information-theoretic private information retrieval. In *Journal of Computer and System Sciences*, 2005.

[BIM00] Amos Beimel, Yuval Ishai, and Tal Malkin. Reducing the Servers' Computation in Private Information Retrieval: PIR with Preprocessing. In *Proceedings of the International Cryptology Conference (CRYPTO)*, 2000.

[BIPW17] Elette Boyle, Yuval Ishai, Rafael Pass, and Mary Wootters. Can we access a database both locally and privately? In *Proceedings of the Theory of Cryptography Conference (TCC)*, 2017.

[CGHK22] Henry Corrigan-Gibbs, Alexandra Henzinger, and Dmitry Kogan. Single-server private information retrieval with sublinear amortized time. In *Proceedings of the International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT)*, 2022.

[CGKS95] Benny Chor, Oded Goldreich, Eyal Kushilevitz, and Madhu Sudan. Private information retrieval. In *Proceedings of the IEEE Symposium on Foundations of Computer Science (FOCS)*, 1995.

[Cha81] David L. Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. In *Communications of the ACM (CACM)*, 1981.

[CHLR18] Hao Chen, Zhicong Huang, Kim Laine, and Peter Rindal. Labeled PSI from fully homomorphic encryption with malicious security. In *Proceedings of the ACM Conference on Computer and Communications Security (CCS)*, October 2018.

[CHR17] Ran Canetti, Justin Holmgren, and Silas Richelson. Towards doubly efficient private information retrieval. In *Proceedings of the Theory of Cryptography Conference (TCC)*, 2017.

[Cle86] Richard Cleve. Limits on the security of coin flips when half the processors are faulty. In *Proceedings of the ACM Symposium on Theory of Computing (STOC)*, 1986.

[CS03] Don Coppersmith and Madhu Sudan. Reconstructing Curves in Three (and Higher) Dimensional Space from Noisy Data. In *Proceedings of the ACM Symposium on Theory of Computing (STOC)*, 2003.

[CSU$^+$19] Albert Cheu, Adam D. Smith, Jonathan R. Ullman, David Zeber, and Maxim Zhilyaev. Distributed differential privacy via shuffling. In *Proceedings of the International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT)*, pages 375–403, 2019.

[CU21] Albert Cheu and Jonathan R. Ullman. The limits of pan privacy and shuffle privacy for learning and estimation. In *Proceedings of the ACM Symposium on Theory of Computing (STOC)*, pages 1081–1094, 2021.

[DG15] Zeev Dvir and Sivakanth Gopi. 2-server pir with sub-polynomial communication. In *Proceedings of the ACM Symposium on Theory of Computing (STOC)*, 2015.

[DMNS06] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating Noise to Sensitivity in Private Data Analysis. In *Proceedings of the Theory of Cryptography Conference (TCC)*, 2006.

[DMR22] Luc Devroye, Abbas Mehrabian, and Tommy Reddad. The total variation distance between high-dimensional gaussians with the same mean, 2022.

[DMS04] Roger Dingledine, Nick Mathewson, and Paul Syverson. Tor: The Second-Generation Onion Router. In *Proceedings of the USENIX Security Symposium*, 2004.

[DN03] Irit Dinur and Kobbi Nissim. Revealing Information while Preserving Privacy. In *Proceedings of the ACM Symposium on Principles of Database Systems (PODS)*, 2003.

[DPC23] Alex Davidson, Gonçalo Pestana, and Sofía Celi. FrodoPIR: Simple, scalable, single-server private information retrieval. In *Proceedings of the Privacy Enhancing Technologies Symposium (PETS)*, 2023.

[EFM$^+$20] Úlfar Erlingsson, Vitaly Feldman, Ilya Mironov, Ananth Raghunathan, Shuang Song, Kunal Talwar, and Abhradeep Thakurta. Encode, shuffle, analyze privacy revisited: Formalizations and empirical evaluation. *CoRR*, abs/2001.03618, 2020.

[GCM$^+$16] Trinabh Gupta, Natacha Crooks, Whitney Mulhern, Srinath Setty, Lorenzo Alvisi, and Michael Walfish. Scalable and Private Media Consumption with Popcorn. In *Proceedings of the USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, 2016.

[GHPS22] Daniel Günther, Maurice Heymann, Benny Pinkas, and Thomas Schneider. GPU-accelerated PIR with Client-Independent Preprocessing for Large-Scale Applications. In *Proceedings of the USENIX Security Symposium*, 2022.

[GIK$^+$24] Adrià Gascón, Yuval Ishai, Mahimna Kelkar, Baiyu Li, Yiping Ma, and Mariana Raykova. Computationally secure aggregation and private information retrieval in the shuffle model. Cryptology ePrint Archive, Paper 2024/870, 2024. `https://eprint.iacr.org/2024/870`. To appear in CCS 2024.

[GK10] Dov Gordon and Jonathan Katz. Partial fairness in secure two-party computation. In *Proceedings of the International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT)*, 2010.

[GKL$^+$20] Paul Grubbs, Anurag Khandelwal, Marie-Sarah Lacharité, Lloyd Brown, Lucy Li, Rachit Agarwal, and Thomas Ristenpart. Pancake: Frequency smoothing for encrypted data stores. In *Proceedings of the USENIX Security Symposium*, 2020.

[GMPV20] Badih Ghazi, Pasin Manurangsi, Rasmus Pagh, and Ameya Velingker. Private Aggregation from Fewer Anonymous Messages. In *Proceedings of the International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT)*, 2020.

[Hen16] Ryan Henry. Polynomial batch codes for efficient IT-PIR. In *Proceedings of the Privacy Enhancing Technologies Symposium (PETS)*, 2016.

[HHCG$^+$23] Alexandra Henzinger, Matthew M. Hong, Henry Corrigan-Gibbs, Sarah Meiklejohn, and Vinod Vaikuntanathan. One Server for the Price of Two: Simple and Fast Single-Server Private Information Retrieval. In *Proceedings of the USENIX Security Symposium*, 2023.

[HSSN$^+$22] Kyle Hogan, Sacha Servan-Schreiber, Zachary Newman, Ben Weintraub, Cristina Nita-Rotaru, and Srinivas Devadas. Shortor: Improving tor network latency via multi-hop overlay routing. In *In Proceedings of the IEEE Symposium on Security and Privacy (S&P)*, 2022.

[IKLM24] Yuval Ishai, Mahimna Kelkar, Daniel Lee, and Yiping Ma. Information-theoretic single-server PIR in the shuffle model. In *Information-Theoretic Cryptography (ITC) 2024*, 2024.

[IKOS04] Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, and Amit Sahai. Batch codes and their applications. In *Proceedings of the ACM Symposium on Theory of Computing (STOC)*, 2004.

[IKOS06] Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, and Amit Sahai. Cryptography from Anonymity. In *Proceedings of the IEEE Symposium on Foundations of Computer Science (FOCS)*, 2006.

[ISN87] Mitsuru Ito, Akira Saito, and Takao Nishizeki. Secret sharing schemes realizing general access structure. In *IEEE Global Telecommunication Conference*, 1987.

[KEB98] Dogan Kesdogan, Jan Egner, and Roland Büschkes. Stop-and-go-mixes providing probabilistic anonymity in an open system. In *Information Hiding*, 1998.

[KO97] Eyal Kushilevitz and Rafail Ostrovsky. Replication is not needed: Single database, computationally-private information retrieval. In *Proceedings of the IEEE Symposium on Foundations of Computer Science (FOCS)*, 1997.

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

[LMW23] Wei-Kai Lin, Ethan Mook, and Daniel Wichs. Doubly efficient private information retrieval and fully homomorphic RAM computation from ring LWE. In *Proceedings of the ACM Symposium on Theory of Computing (STOC)*, 2023.

[LYK$^+$19] Donghang Lu, Thomas Yurek, Samarth Kulshreshtha, Rahul Govind, Aniket Kate, and Andrew K. Miller. Honeybadgermpc and asynchromix: Practical asynchronous MPC and its application to anonymous communication. In *Proceedings of the ACM Conference on Computer and Communications Security (CCS)*, pages 887–903, 2019.

[MNS09] Tal Moran, Moni Naor, and Gil Segev. An optimally fair coin toss. In *Proceedings of the Theory of Cryptography Conference (TCC)*, 2009.

[MR23] Muhammad Haris Mughees and Ling Ren. Vectorized Batch Private Information Retrieval. In *In Proceedings of the IEEE Symposium on Security and Privacy (S&P)*, 2023.

[MW22] Samir Jordan Menon and David J. Wu. Spiral: Fast, High-Rate Single-Server PIR via FHE Composition. In *In Proceedings of the IEEE Symposium on Security and Privacy (S&P)*, 2022.

[TDG16] Raphael R. Toledo, George Danezis, and Ian Goldberg. Lower-Cost ϵ-Private Information Retrieval. In *Proceedings of the Privacy Enhancing Technologies Symposium (PETS)*, 2016.

[vdHLZZ15] Jelle van den Hoof, David Lazar, Matei Zaharia, and Nickolai Zeldovich. Vuvuzela: Scalable Private Messaging Resistant to Traffic Analysis. In *Proceedings of the ACM Symposium on Operating Systems Principles (SOSP)*, 2015.

[WY05] David Woodruff and Sergey Yekhanin. A Geometric Approach to Information-Theoretic Private Information Retrieval. In *Computational Complexity Conference (CCC)*, 2005.

[ZKP16] Yupeng Zhang, Jonathan Katz, and Charalampos Papamanthou. All your queries are belong to us: The power of file-injection attacks on searchable encryption. In *Proceedings of the USENIX Security Symposium*, 2016.

[ZPSZ23] Mingxun Zhou, Andrew Park, Elaine Shi, and Wenting Zheng. Piano: Extremely simple, single-server pir with sublinear server computation, 2023.

A Deferred Material for Warm-Up Results (Section 5)

Now we provide formal analysis of why using 2 additive shares is not enough for security. Recall that the query indices are encoded in the space $\mathbb{F}_2^m$ for some m. Let $i \in [n]$ be the index such that its encoding is 0^m and let $i' \in [n]$ be the index such that its encoding is 1^m. Now consider two sets of query indices, $I = (i, \ldots, i)$ and $I' = (i', \ldots, i')$. Note that a 1 can only be split as $0 + 1$, but a 0 can be split to either as $0 + 0$ or $1 + 1$. Consider first, the extreme case where $m = 1$; here, for I', there will always be exactly the same number of 0s and 1s while this is not true for I. A similar issue also exists for a general m; if one share is $\beta \in \mathbb{F}_2^m$, then for I', there will be a corresponding share $1^m - \beta$ (for instance if one share is $(0, 1, 0, 1)$, the other share will be $(1, 0, 1, 0)$). This means that the total number of shares that are β will always be equal to the total number of shares that are $1^m - \beta$. The same will not hold for I, which allows the server to distinguish between I and I'.

We show that there exists a constant-advantage distinguisher that can tell between case 0: 1^m; and case 1: queries drawn uniformly and iid from $\mathbb{Z}_q$, when only constant number of noise queries per client is added.

To start off, we provide an important observation when no noise is added. For simplicity, we only consider the case where q is even, although the results extend, with slight modification, to when q is odd. First, we observe that when 1 is split into two shares over $\mathbb{Z}_q$ into $a + b = 1$, there must always be one share which is in $S_L := 1, \ldots, \frac{q}{2}$, and the other share in $S_H := \frac{q}{2} + 1, \ldots, q$ (aka 0). Let Q_L and Q_H be random variables for the number of queries landing in S_L and S_H respectively, and $\Delta = Q_L - Q_H$. By the previous observation, $Q_L = Q_H$ in every 2 additive sharing of 1^m. However, in the uniform case, each query has equal probability of ending up in S_L or S_H, independently of each other; therefore, Δ is exactly a $Q = 2C$ length random walk (where C is the number of clients/queries, and Q the number of shares). The probability of said random walk staying at 0 goes to 0 as C increases. And so a simple distinguisher (with advantage almost 1) outputs 0 if $\Delta = 0$, and 1 otherwise.

What happens when we add noise queries to the mixed shares? These noise queries each contribute 2 uniform, *i.i.d.* balls to our buckets S_L and S_H; therefore, they serve to extend the random walk, Δ. If we add k noise queries per client, then Δ in case 0 is a $2kC = kQ$ length random walk, while in case 1, Δ is a $2(k+1)C = (k+1)Q$ length random walk.

It remains to lower bound the total variation distance between $\mathrm{RW}(kQ)$ and $\mathrm{RW}((k+1)Q)$. Since we are interested in asymptotics, it suffices to instead lower bound the total variation distance between $\mathcal{N}(0, kQ)$ and $\mathcal{N}(0, (k+1)Q)$. Applying Theorem 1.1 from [DMR22], we know that this distance is lower bounded by $\frac{1}{100k}$.

B Imperfect Shuffling

While we work primarily with perfect shufflers, we show a simple result here that highlights the robustness of our constructions to imperfect shufflers.

Definition B.1 (Imperfect Shuffler). A shuffler $\Pi = \{\Pi_c\}_{c \in \mathbb{N}}$ where each Π_c is a distribution over the symmetric group $\mathfrak{S}_c$ is said to be at most ζ-imperfect if for all c,

$$\max_{X \sim \Pi_c} \Pr[X = \sigma] \leq \zeta \cdot \Pr_{Y \sim \widetilde{\Pi}_c} [Y = \sigma]$$

where $\widetilde{\Pi} = \{\widetilde{\Pi}_c\}_{c\in\mathbb{N}}$ is the uniform shuffler. In other words, a ζ-imperfect allows for the probability of any particular shuffling to be at most a factor of ζ larger than the uniform case.

We now illustrate the robustness of the constructions to imperfect shuffling. In particular, if Π' is ζ-imperfect for a constant ζ, then the statistical distance of our construction when Π' is used is at most ζ times the statistical distance when $\widetilde{\Pi}$ is used. The following lemma shows this more generically:

Lemma B.2. *Consider any distributions $\mathcal{D}_\Pi$ and $\mathcal{D}'_\Pi$ that depend on a shuffler Π on group $\mathbb{G}$. Let $\widetilde{\Pi}$ be the uniform shuffler on group $\mathbb{G}$ and Π' be a ζ-imperfect shuffler. Then, $\mathsf{SD}(\mathcal{D}_{\Pi'}, \mathcal{D}'_{\Pi'}) \leq \zeta \cdot \mathsf{SD}(\mathcal{D}_{\widetilde{\Pi}}, \mathcal{D}'_{\widetilde{\Pi}})$.*

Proof.

$$\begin{aligned}
\mathsf{SD}(\mathcal{D}_{\Pi'}, \mathcal{D}'_{\Pi'}) &= \sum_\sigma \Pr[\Pi' = \sigma] \cdot \mathsf{SD}((\mathcal{D}_{|\sigma}, \mathcal{D}'_{|\sigma})) \\
&\leq \sum_\sigma \zeta \cdot \Pr[\widetilde{\Pi} = \sigma] \cdot \mathsf{SD}((\mathcal{D}_{|\sigma}, \mathcal{D}'_{|\sigma})) \\
&= \zeta \sum_\sigma \Pr[\widetilde{\Pi} = \sigma] \cdot \mathsf{SD}((\mathcal{D}_{|\sigma}, \mathcal{D}'_{|\sigma})) \\
&= \zeta \cdot \mathsf{SD}(\mathcal{D}_{\widetilde{\Pi}}, \mathcal{D}'_{\widetilde{\Pi}})
\end{aligned}$$

□

C Complete Security Proof for Add-ShPIR (Theorem 6.1)

We now provide the full details of the security proof for Add-ShPIR—our generic composition that uses any k-server PIR as OPIR and 2-additive PIR as IPIR. Recall that our proof outline consists of three major steps; the subsequent subsections formally describe each of these steps.

C.1 Bounding the OPIR Edit Distance (Proof of Lemma 6.2)

We start with the details for our first major proof step, namely bounding the edit distance between the OPIR sub-queries. This only requires proving Lemma 6.2, which we do below.

Proof. When balls are thrown according to the distribution $\mathcal{B}$, define $\mathbb{U}_\alpha$ to be the random variable for the number of balls thrown into bin α, and $\mathbb{U}_{b,\alpha}$ to be the indicator variable that is 1 exactly when the b^{th} ball is thrown into bin α and 0 otherwise.

Note that $\mathbb{U}_{b,\alpha}$ and $\mathbb{U}_{b',\alpha}$ are independent when $b \neq b'$ the balls are thrown in a pairwise independent fashion. Now, each $\mathbb{U}_{b,\alpha}$ is a Bernoulli random variable with parameter $1/N$. Therefore, $\mathbb{E}[\mathbb{U}_{b,\alpha}] = \frac{1}{N}$ and $\mathrm{Var}[\mathbb{U}_{b,\alpha}] = \frac{1}{N}\left(1 - \frac{1}{N}\right)$.

Now, by linearity of expectation, for all bins α, we have $\mathbb{E}[\mathcal{U}_\alpha] = B/N$. Furthermore, since the balls are thrown in a pairwise independent way, the variance is also linear, and therefore, $\mathrm{Var}[\mathbb{U}_\alpha] = \sum_{b=1}^{B} \mathrm{Var}[\mathbb{U}_{b,\alpha}] = \frac{B}{N} \cdot \left(1 - \frac{1}{N}\right)$. Therefore,

$$\mathbb{E}[\mathbb{U}_\alpha^2] = \mathrm{Var}[\mathbb{U}_\alpha] + (\mathbb{E}[\mathbb{U}_\alpha])^2 = \frac{BN - B + B^2}{N^2}.$$

Similarly define $\mathbb{V}_\alpha$ and $\mathbb{V}_{b,\alpha}$ when the balls are thrown according to distribution $\mathcal{B}'$. Note that all the above analysis also carries over for $\mathbb{V}_\alpha$. Now,

$$\begin{aligned}
\mathbb{E}[|\mathbb{U}_\alpha - \mathbb{V}_\alpha|] &\leq \sqrt{\mathbb{E}[|\mathbb{U}_\alpha - \mathbb{V}_\alpha|^2]} = \sqrt{\mathbb{E}[\mathbb{U}_\alpha^2 + 2\mathbb{U}_\alpha\mathbb{V}_\alpha + \mathbb{V}_\alpha^2]} \\
&= \sqrt{\mathbb{E}[\mathbb{U}_\alpha^2] + 2\mathbb{E}[\mathbb{U}_\alpha]\mathbb{E}[\mathbb{V}_\alpha] + \mathbb{E}[\mathbb{V}_\alpha^2]} \\
&= \sqrt{\frac{2BN - 2B}{N^2}} \leq \sqrt{\frac{2B}{N}}.
\end{aligned}$$

where the first inequality is from the fact that $(\mathbb{E}[\mathbb{X}])^2 \leq \mathbb{E}[\mathbb{X}^2]$ for any random variable $\mathbb{X}$, and the third step is from linearity of expectation and the fact that $\mathbb{U}_\alpha$ and $\mathbb{V}_\alpha$ are independent.

Finally using the linearity of expectation again, we can compute the expected edit distance as:

$$\mathbb{E}_{\mathbf{u}\sim\mathcal{B},\mathbf{v}\sim\mathcal{B}'}[\mathsf{ED}(\mathbf{u},\mathbf{v})] = \frac{1}{2}\mathbb{E}\left[\sum_\alpha |\mathbb{U}_\alpha - \mathbb{V}_\alpha|\right] \leq \frac{N}{2}\sqrt{\frac{2B}{N}} = \sqrt{\frac{BN}{2}}.$$

□

C.2 Bounding the Edit Distance of IPIR Shares

We now provide details for our second major proof step on bounding the edit distance between the IPIR shares. We start by showing that when looking at the final statistical distance, it is enough to only consider parts that differ between $\mathbf{u}$ and $\mathbf{v}$. In particular, we prove the following statement.

Lemma C.1. *Consider $\ell \geq 0$ amd two (B, N)-valid configurations $\mathbf{u}$ and $\mathbf{v}$. Then,*

$$\mathsf{SD}(\mathsf{Share}^\ell_{\mathbf{u}}, \mathsf{Share}^\ell_{\mathbf{v}}) \leq \mathsf{SD}(\mathsf{Share}^\ell_{\mathbf{u}\ominus\mathbf{v}}, \mathsf{Share}^\ell_{\mathbf{v}\ominus\mathbf{u}})$$

Proof. Let $f_{\mathbf{u}}, f_{\mathbf{v}}, f_{\mathbf{u}\ominus\mathbf{v}}, f_{\mathbf{v}\ominus\mathbf{u}}$ denote the probability mass functions of $\mathsf{Share}^k_{\mathbf{u}}, \mathsf{Share}^\ell_{\mathbf{v}}, \mathsf{Share}^\ell_{\mathbf{u}\ominus\mathbf{v}}$ and $\mathsf{Share}^\ell_{\mathbf{v}\ominus\mathbf{u}}$ respectively. Define $\mathbf{c} = \mathbf{u} \sqcap \mathbf{v}$ and let $f_{\mathbf{c}}$ be the probability mass function of $\mathsf{Share}^\ell_{\mathbf{c}}$. Now,

$$\begin{aligned}
\mathsf{SD}(\mathsf{Share}^\ell_{\mathbf{u}}, \mathsf{Share}^\ell_{\mathbf{v}}) &= \frac{1}{2}\sum_{\mathbf{w}} |f_{\mathbf{u}}(\mathbf{w}) - f_{\mathbf{v}}(\mathbf{w})| \\
&= \frac{1}{2}\sum_{\mathbf{w}} \left| \left(\sum_{\mathbf{w}' \leq \mathbf{w}} f_{\mathbf{c}}(\mathbf{w} \ominus \mathbf{w}') f_{\mathbf{u}\ominus\mathbf{v}}(\mathbf{w}') \right) - \left(\sum_{\mathbf{w}' \leq \mathbf{w}} f_{\mathbf{c}}(\mathbf{w} \ominus \mathbf{w}') f_{\mathbf{v}\ominus\mathbf{u}}(\mathbf{w}') \right) \right|
\end{aligned}$$

by marginalization and since $\mathbf{w}'$ and $\mathbf{w} \ominus \mathbf{w}'$ deal with separate initial balls which would make their sharing independent. We now get,

$$
\begin{aligned}
\mathsf{SD}(\mathsf{Share}^{\ell}_{\mathbf{u}}, \mathsf{Share}^{\ell}_{\mathbf{v}}) &= \frac{1}{2}\sum_{\mathbf{w}}\left|\sum_{\mathbf{w}'\leq\mathbf{w}} f_{\mathbf{c}}(\mathbf{w}\ominus\mathbf{w}')\left(f_{\mathbf{u}\ominus\mathbf{v}}(\mathbf{w}') - f_{\mathbf{v}\ominus\mathbf{u}}(\mathbf{w}')\right)\right| \\
&\leq \frac{1}{2}\sum_{\mathbf{w}}\sum_{\mathbf{w}'\leq\mathbf{w}} f_{\mathbf{c}}(\mathbf{w}\ominus\mathbf{w}')\left|f_{\mathbf{u}\ominus\mathbf{v}}(\mathbf{w}') - f_{\mathbf{v}\ominus\mathbf{u}}(\mathbf{w}')\right| \\
&= \frac{1}{2}\sum_{\mathbf{w}'}\left(\left|f_{\mathbf{u}\ominus\mathbf{v}}(\mathbf{w}') - f_{\mathbf{v}\ominus\mathbf{u}}(\mathbf{w}')\right|\sum_{\mathbf{w}\geq\mathbf{w}'} f_{\mathbf{c}}(\mathbf{w}\ominus\mathbf{w}')\right) \\
&\leq \frac{1}{2}\sum_{\mathbf{w}'}\left|f_{\mathbf{u}\ominus\mathbf{v}}(\mathbf{w}') - f_{\mathbf{v}\ominus\mathbf{u}}(\mathbf{w}')\right|\cdot 1 \\
&= \mathsf{SD}(\mathsf{Share}^{\ell}_{\mathbf{u}\ominus\mathbf{v}}, \mathsf{Share}^{\ell}_{\mathbf{v}\ominus\mathbf{u}})
\end{aligned}
$$

□

This allows us to restrict our attention to only $\mathbf{u}\ominus\mathbf{v}$ and $\mathbf{v}\ominus\mathbf{u}$ which are (δ, B)-valid configuration. We will now find the edit distance after splitting each of the δ balls into two additive shares. Formally, we show the following lemma:

Lemma C.2. *Consider two (δ, N)-valid configurations $\mathbf{u}$ and $\mathbf{v}$. Then,*

$$\mathbb{E}[\mathsf{ED}(\mathsf{Share}_{\mathbf{u}}, \mathsf{Share}_{\mathbf{v}}) \leq \sqrt{2\delta N}.$$

Proof. When balls are thrown according to the distribution $\mathsf{Share}_{\mathbf{u}}$, define $\mathbb{U}_\alpha$ as the random variable for the number of balls thrown into bin α. Define $\mathbb{V}_\alpha$ for distribution $\mathsf{Share}_{\mathbf{v}}$. First observe by linearity of expectation that:

$$\mathbb{E}[\mathsf{ED}(\mathsf{Share}_{\mathbf{u}}, \mathsf{Share}_{\mathbf{v}})] = \frac{1}{2}\sum_{\alpha}\mathbb{E}\left[|\mathbb{U}_\alpha - \mathbb{V}_\alpha|\right]$$

Now, to find the distribution of $\mathbb{U}_\alpha$, we need to find when additively splitting a ball results in an addition to the bin α. Let $(b_1, \ldots, b_\delta)$ denote the vector of balls in $\mathbf{u}$, and define $\mathbb{U}_{i,\alpha}$ to be the number of additive shares of ball b_i that go into bin α. Observe that for any particular α, all $\mathbb{U}_{i,\alpha}$ are independent and that $\mathbb{U}_\alpha = \sum_{i=1}^{\delta}\mathbb{U}_{i,\alpha}$. Now, consider two cases for each ball b_i, and a bin α:

1. $b_i = \alpha + \alpha$ (in the group $\mathbb{G}$). In this case, if the first additive share of b_i is sampled as α, then both additive shares will go into bin α; otherwise no share will go into bin α. This means that $\mathbb{U}_{i,\alpha} \sim 2\cdot\mathsf{Ber}(1/N)$.

2. $b_i \neq \alpha + \alpha$ (in the group $\mathbb{G}$). In this case, if the first additive share of b_i is sampled either as α or $b_i - \alpha$, then exactly one of the additive shares will go into bin α; otherwise no share will go into bin α. This means that $\mathbb{U}_{i,\alpha} \sim \mathsf{Ber}(2/N)$.

Assume that there are $\lambda_{\mathbf{u},\alpha}$ balls that satisfy the first case and $\delta - \lambda_{\mathbf{u},\alpha}$ balls that satisfy the second case. Using the fact that the $\mathbb{U}_{i,\alpha}$ are independent, we can now compute the distribution of $\mathbb{U}_\alpha$ as:

$$\mathbb{U}_\alpha \sim 2\cdot\mathsf{Binomial}(\lambda_{\mathbf{u},\alpha}, 1/N) + \mathsf{Binomial}(\delta - \lambda_{\mathbf{u},\alpha}, 2/N).$$

Consequently, the following hold:

$$\mathbb{E}[\mathbb{U}_\alpha] = \frac{2\lambda_{\mathbf{u},\alpha}}{N} + \frac{2(\delta - \lambda_{\mathbf{u},\alpha})}{N} = \frac{2\delta}{N}.$$

$$\mathrm{Var}[\mathbb{U}_\alpha] = 4\lambda_{\mathbf{u},\alpha}\frac{N-1}{N^2} + (\delta - \lambda_{\mathbf{u},\alpha})\frac{2(N-2)}{N^2} = \frac{2N\lambda_{\mathbf{u},\alpha} + 2N\delta - 4\delta}{N^2}.$$

Similarly, we can compute

$$\mathbb{E}[\mathbb{V}_\alpha] = \frac{2\delta}{N} \qquad \text{and} \qquad \mathrm{Var}[\mathbb{V}_\alpha] = \frac{2N\lambda_{\mathbf{v},\alpha} + 2N\delta - 4\delta}{N^2}.$$

where $\lambda_{\mathbf{v},\alpha}$ is the number of balls in $\mathbf{v}$ that are equal to $\alpha + \alpha$ (in group $\mathbb{G}$). Now applying the Jensen's inequality $\mathbb{E}[Z] \le \sqrt{\mathbb{E}[Z^2]}$ to the random variable $|\mathbb{U}_\alpha - \mathbb{V}_\alpha|$, we get:

$$\begin{aligned}
\mathbb{E}[|\mathbb{U}_\alpha - \mathbb{V}_\alpha|] &\le \sqrt{\mathbb{E}[(\mathbb{U}_\alpha - \mathbb{V}_\alpha)^2]} = \sqrt{\mathbb{E}[(\mathbb{U}_\alpha)^2] + \mathbb{E}[(\mathbb{V}_\alpha)^2] - 2\cdot\mathbb{E}[\mathbb{U}_\alpha]\cdot\mathbb{E}[\mathbb{U}_\alpha]} \\
&= \sqrt{\frac{2N\lambda_{\mathbf{u},\alpha} + 2N\delta - 4\delta + 4\delta^2}{N^2} + \frac{2N\lambda_{\mathbf{v},\alpha} + 2N\delta - 4\delta + 4\delta^2}{N^2} - \frac{8\delta^2}{N^2}} \\
&= \sqrt{\frac{2\lambda_{\mathbf{u},\alpha} + 2\lambda_{\mathbf{v},\alpha} + 4\delta}{N} - \frac{8\delta}{N^2}} \\
&\le \sqrt{\frac{2\lambda_{\mathbf{u},\alpha} + 2\lambda_{\mathbf{v},\alpha} + 4\delta}{N}} \\
&\le \sqrt{\frac{8\delta}{N}}
\end{aligned}$$

since $0 \le \lambda_{\mathbf{u},\alpha}, \lambda_{\mathbf{v},\alpha} \le \delta$ (in fact, we have $\sum_\alpha \lambda_{\mathbf{u},\alpha} \le \delta$). Therefore, we can now compute the edit distance as follows:

$$\mathsf{ED}(\mathsf{Share}_{\mathbf{u}}, \mathsf{Share}_{\mathbf{v}}) = \frac{1}{2}\sum_\alpha \mathbb{E}[|\mathbb{U}_\alpha - \mathbb{V}_\alpha|] \le \frac{N}{2}\cdot\sqrt{\frac{8\delta}{N}} = \sqrt{2\delta N}.$$

□

Combining this with the result from Lemma 6.2, we can now compute the expected edit distance when $\mathbf{u}$ and $\mathbf{v}$ follow a distribution instead of being fixed.

$$\begin{aligned}
\mathbb{E}_{\mathbf{u}\sim\mathcal{B},\mathbf{v}\sim\mathcal{B}'}\left[\mathsf{ED}(\mathsf{Share}_{\mathbf{u}\ominus\mathbf{v}}, \mathsf{Share}_{\mathbf{v}\ominus\mathbf{u}})\right] &\le \sqrt{2N}\cdot\mathbb{E}_{\mathbf{u}\sim\mathcal{B},\mathbf{v}\sim\mathcal{B}'}\left[\sqrt{\mathsf{ED}(\mathbf{u},\mathbf{v})}\right] \\
&\le \sqrt{2N}\left(\frac{BN}{2}\right)^{1/4} = (2)^{1/4}B^{1/4}(N)^{3/4}
\end{aligned}$$

where the second step is by the concave Jensen's inequality.

C.3 Bounding the Final Statistical Distance

Before we bound the final statistical distance, we introduce a useful result from Boyle et al. [BGIK22].

Lemma C.3 ([BGIK22]). *Consider ℓ balls thrown into N bins (labeled using $[N]$ without loss of generality) independently and uniformly at random. Let $\mathcal{U}_\alpha$ denote the final distribution of the configuration after another ball is added into bin α. Then, for all bins α and α', $\mathsf{SD}(\mathcal{U}_\alpha, \mathcal{U}_{\alpha'}) \leq \sqrt{\frac{N}{\ell}}$.*

While the original result in [BGIK22] is stated in terms of removing a ball either from bin α or α', we note that our formulation is equivalent since the statistical distance does not change by adding the same balls (one each in the two bins) to both distributions.

A more general bound for adding δ balls can also easily be derived. Suppose that we use the notation $S_\ell(\delta)$ to denote the maximum statistical distance when δ balls are added after throwing ℓ balls independently and uniformly at random. In particular, for $\Upsilon = (\upsilon_1, \dots, \upsilon_\delta) \in [N]^\delta$, let $\mathcal{U}_\Upsilon$ denote the distribution when after throwing ℓ balls, a ball is added to each bin υ_i; Then $S_\ell(\delta) = \max_{\Upsilon, \Upsilon'} \mathsf{SD}(\mathcal{U}_\Upsilon, \mathcal{U}_{\Upsilon'})$. By hopping one ball at a time, we can use Lemma C.3 to directly conclude that $S_\ell(\delta) \leq \delta \cdot \sqrt{N/\ell}$.

We are now ready to bound the final statistical distance. Applying Markov's inequality to the result from the previous section, we get:

$$\Pr_{\mathbf{u} \sim \mathcal{B}, \mathbf{v} \sim \mathcal{B}'} \left[\mathsf{ED}(\mathsf{Share}_{\mathbf{u} \ominus \mathbf{v}}, \mathsf{Share}_{\mathbf{v} \ominus \mathbf{u}}) \geq \gamma \cdot B^{1/4+\tau} N^{1/8} \right] \leq \frac{\sqrt[4]{2} \cdot B^{1/4} N^{3/4}}{\gamma \cdot B^{1/4+\tau} N^{1/8}} \leq \frac{2 \cdot N^{5/8}}{\gamma \cdot B^\tau}.$$

Define this probability as ρ_τ. As the final step, we can now use Lemma C.1 and Lemma C.3 to compute the final statistical distance. Using $\tau = 1/8$ and $\ell = \gamma^4 B$, we get:

$$\begin{aligned} \mathsf{SD}(\mathsf{Share}^\ell_{\mathbf{u}}, \mathsf{Share}^\ell_{\mathbf{v}}) &\leq \mathsf{SD}(\mathsf{Share}^\ell_{\mathbf{u} \ominus \mathbf{v}}, \mathsf{Share}^\ell_{\mathbf{v} \ominus \mathbf{u}}) \\ &\leq (1 - \rho_\tau) \cdot S_\ell(B^{1/4+\tau} N^{1/8}) + \rho_\tau \cdot S_\ell(B) \\ &\leq 1 \cdot \gamma B^{1/4+\tau} N^{1/8} \cdot \sqrt{\frac{N}{\ell}} + \frac{2 \cdot N^{5/8}}{\gamma B^\tau} \cdot 1 \\ &= \frac{N^{5/8}}{\gamma B^{1/4-\tau}} + \frac{2 \cdot N^{5/8}}{\gamma B^\tau} \leq \frac{3 \cdot N^{5/8}}{\gamma B^{1/8}}. \end{aligned}$$

Casting this back to our PIR context, since we have $N = |\mathcal{Q}_{\mathsf{IPIR}}|$ bins and $B = kC$ balls, and $\ell = B$, the final statistical distance is bounded by $\frac{3|\mathcal{Q}_{\mathsf{IPIR}}|^{5/8}}{(kC)^{1/8}}$. Recall that in Section 6.1.1, we can make $|\mathcal{Q}_{\mathsf{IPIR}}| = \Theta(n)$, where n is the database size. Therefore, for all $\epsilon = \epsilon(n) \geq 0$, there exists a constant d such that given $C \geq \frac{1}{\epsilon^8} \cdot \frac{dn^5}{k}$ honest clients queries, the statistical distance is bounded by ϵ.

Notice that there is an interesting trade-off between the number of clients required and the random noise used per client. By having each client provide γ times more noise queries, the statistical distance is reduced by a factor of $\sqrt[4]{\gamma}$, which in turn reduces the number of clients required by a factor of $(\sqrt[4]{\gamma})^8 = \gamma^2$.

C.4 Cost Analysis

As in Theorem 6.1, k, Q, A are all functions of n. Below we write e.g., $k(n)$ as k for simplicity. To analyze the cost, we need to first analyze the size of x' in Construction 6.1. Let σ be the size of x'.

- When the OPIR servers have different Answer algorithms, the IPIR database x' has $k \cdot Q$ entries, each of A bits. Here $\sigma = kQ$.

- When the OPIR servers have the same Answer algorithm, the size x' is simply Q, each entry of x' is of A bits. Here $\sigma = Q$.

Per-query communication. To issue a query to the original n-bit database, the client sends $3k$ messages in total ($2k$ messages for shares of OPIR sub-queries and k dummies). Each message is an IPIR sub-query, therefore the query size for ShPIR is $O(k \cdot \log \sigma)$, and the answer size is $O(kA\sigma^{1/2})$. The communication cost is dominated by the answer size, hence $O(kA\sigma^{1/2})$. When $\sigma = kQ$, the communication is $O(k^{3/2} \cdot A \cdot Q^{1/2})$; when $\sigma = Q$, the communication is $O(k \cdot A \cdot Q^{1/2})$.

Per-query computation. Assuming preprocessing, the server computation is the number of bits it reads, which is simply the answer size. So the computation is the same as above.

Server storage. To preprocess a size-σ database in the two-server additive PIR protocol (Figure 4.1), the server chooses the parameter m' for IPIR and a constant c such that $m' = c \cdot \log \sigma$. So the sub-query space of IPIR, namely $\mathcal{Q}_{\mathsf{IPIR}}$, has size $2^{m'} = \sigma^c$ (and consequently the number of entries in the lookup table). We can in fact use a more fine-grained choice of m', so that the size of $\mathcal{Q}_{\mathsf{IPIR}}$ is $\widetilde{O}(\sigma)$; we provide details in Appendix E.1.

Each entry in the lookup table is an answer polynomial with the number of bits $A\sigma^{1/2}$. Putting these together, the server storage, including the preprocessing bits, is $\widetilde{O}(A \cdot \sigma^{3/2})$. If $\sigma = kQ$, then the storage is $\widetilde{O}(A \cdot k^{3/2} \cdot Q^{3/2})$; if $\sigma = Q$, then the storage is $\widetilde{O}(A \cdot Q^{3/2})$.

D Complete Security Proof for s-CNF-ShPIR (Theorem 6.3)

We now provide details for analyzing the construction where CNF-sharing is used for IPIR instead of 2-additive sharing. The basic structure of the proof is quite similar; notice that among the three major proof steps for Theorem 6.1, only the second part needs to be changed to reflect the CNF-sharing. This essentially requires analysis on how the balls in a configuration $\mathbf{u}$ get split into new balls corresponding to the CNF shares.

Definition D.1 (Cyclic rotations)**.** For a vector $\alpha = (\alpha_1, \dots, \alpha_s)$, define its γ-cyclic rotation $(0 \le \gamma < s)$ as the vector $\alpha^{(\gamma)} = (\alpha_{\gamma+1}, \dots, \alpha_s, \alpha_1, \dots, \alpha_\gamma)$ where α_0 is defined to be α_s.

CNF-sharing details. Consider a (δ, N)-valid configuration $\mathbf{u}$ where the bins are labeled using elements in $\mathbb{G}$. For a given ball b, the s-CNF sharing procedure is as follows: First b is randomly split into s additive shares $\beta = (\beta_1, \dots, \beta_s)$; i.e., $\beta_1, \dots, \beta_{s-1}$ are first independently and uniformly sampled from $\mathbb{G}$, and then β_s is set to $b - \sum_{i=0}^{s-1} \beta_i$. Now, the s-CNF shares are defined to the cyclic rotations of β where the last element is dropped. In particular, the CNF-shares of β are $\alpha^{(0)}, \dots, \alpha^{(s-1)}$ where $\alpha^{(i)} = (\beta_{i+1}, \dots, \beta_s, \beta_1, \dots, \beta_{i-1})$ and β_0 is defined to β_s.

D.1 Balls-and-Bins Analysis for CNF-shares

Notice that CNF-share is a vector in $\mathbb{G}^{s-1}$, and consequently, there are N^{s-1} bins within which the ball corresponding to each CNF-share can lie. Our goal now, very abstractly, is to understand the conditions under which one (or more) of the s balls corresponding to the CNF-shares resultant from splitting a ball b in $\mathbf{u}$ fall into a particular bin $\alpha \in \mathbb{G}^{s-1}$. This involves taking into account the symmetries of the CNF-shares towards which, we introduce some useful definitions.

Definition D.2 (Cyclic symmetries)**.** For a vector $\alpha = (\alpha_1, \ldots, \alpha_s)$, define the number of cyclic symmetries of α, denoted by $\mathsf{SymCyc}(\alpha)$, as the number of cyclic rotations $\alpha^{(\gamma)}$ where $(0 \leq \gamma < s)$ that are equal to α. Further, define the number of distinct cyclic rotations, denoted by $\mathsf{DistCyc}(\alpha)$, as the cardinality of the set $\{\alpha^{(\gamma)} \mid 0 \leq \gamma < s\}$.

Lemma D.3. *For any $\alpha = (\alpha_1, \ldots, \alpha_s)$, it holds that $\mathsf{SymCyc}(\alpha) \cdot \mathsf{DistCyc}(\alpha) = s$.*

Proof. The proof is quite straightforward using a group theoretic formulation. Notice that the group of cyclic rotations of is isomorphic to the group $\mathbb{Z}_s$ under addition modulo s; intuitively $\gamma \in \mathbb{Z}_s$ will correspond to a γ-cyclic rotation. Let c be the smallest positive integer such that $\alpha^{(c \bmod s)} = \alpha$. Then for all $\alpha^{(\gamma)} = \alpha$, notice that $\gamma \in \langle c \rangle$ (the subgroup of $\mathbb{Z}_s$ generated by c) which is therefore of size exactly $\mathsf{SymCyc}(\alpha)$. Further, the number of cosets of $\langle c \rangle$ in $\mathbb{Z}_s$ is exactly the number of distinct cyclic rotations $\mathsf{DistCyc}(\alpha)$. Therefore, by Lagrange's theorem, we directly have $\mathsf{SymCyc}(\alpha) \cdot \mathsf{DistCyc}(\alpha) = s$. ☐

Now, coming back to the CNF-sharing problem at hand, consider a (δ, N)-valid configuration $\mathbf{u}$. Define s-CNF-$\mathsf{Share}_{\mathbf{u}}$ to be the distribution of the balls-and-bins configuration when each ball in $\mathbf{u}$ is split into s-CNF shares. Note that we only need to consider (δ, N)-configurations since the proof of Lemma C.1 also directly works for s-CNF-Share. We now show the following lemma.

Lemma D.4. *Consider a (δ, N)-valid configurations $\mathbf{u}$ and $\mathbf{v}$. Then,*

$$\mathsf{ED}(s\text{-}\mathsf{CNF}\text{-}\mathsf{Share}_{\mathbf{u}}, s\text{-}\mathsf{CNF}\text{-}\mathsf{Share}_{\mathbf{v}}) \leq sN^{(s-1)/2}\sqrt{\delta}.$$

Proof. Analyzing s-CNF-$\mathsf{Share}_{\mathbf{u}}$ essentially boils down to two parts:

1. What is the probability that that a ball b will lead to a CNF-share α (or equivalently, a ball in bin α within s-CNF-$\mathsf{Share}_{\mathbf{u}}$) (notice that the random variable will be $\mathsf{Bernoulli}$ and so we only need to find the probability).

2. Due to the symmetries of CNF-sharing, when one CNF-share is α, how many more shares will also be exactly α? In other words, if a ball lands in bin α, does this force any other balls to also land in α? (for instance, in the 2-additive sharing, when we had $b = 2\alpha$ for a ball b and bin α, if one additive share was α, then the other share would also be α).

Let $\mathbb{U}_\alpha$ represent the random variable for the number of balls in bin α for the distribution s-CNF-$\mathsf{Share}_{\mathbf{u}}$. As in the proof for Theorem 6.4, we wish to find $\mathbb{E}[\mathbb{U}_\alpha]$ and $\mathrm{Var}[\mathbb{U}_\alpha]$ and use them to bound the edit distance between s-CNF-$\mathsf{Share}_{\mathbf{u}}$ and s-CNF-$\mathsf{Share}_{\mathbf{v}}$ for any two $\mathbf{u}$ and $\mathbf{v}$.

For $1 \leq \tau \leq s$ and $\alpha \in \mathbb{G}^{s-1}$, let $\lambda_{\mathbf{u},\tau,\alpha}$ denote the number of balls b_i in $\mathbf{u}$ such that for the vector $\alpha^* = (\alpha_1, \ldots, \alpha_{s-1}, b_i - \sum_i \alpha_i)$, it holds that $\mathsf{SymCyc}(\alpha^*) = \tau$, and consequently $\mathsf{DistCyc}(\alpha^*) = s/\tau$ (from Lemma D.3). First notice that $\sum_\tau \lambda_{\mathbf{u},\tau,\alpha} = \delta$ since each ball will in some $\mathsf{SymCyc}(\alpha)$ value from 0 to s.

Now, $\mathsf{SymCyc}(\alpha^*)$ exactly corresponds to the number of CNF-shares that will fall into bin α if one CNF-share for the ball b_i is α. In addition, the probability that a CNF-share for b_i is α is exactly the number of distinct cyclic rotations divided by the number of bins, i.e., $\frac{\mathsf{DistCyc}(\alpha^*)}{N^{s-1}}$.

The number of balls added to bin α by each CNF-share of a ball in $\mathbf{u}$ is a Bernoulli random variable with probability $\frac{\mathsf{DistCyc}(\alpha^*)}{N^{s-1}}$; $\mathbb{U}_\alpha$ is just the sum of all these Bernoulli random variables. However, the symmetries of the CNF-sharing will create dependence between these random variables; this happens exactly for $\mathsf{SymCyc}(\alpha^*)$ number of variables, leading to their sum being distributed as $\mathsf{SymCyc}(\alpha^*)\mathsf{Ber}\cdot\left(\frac{\mathsf{DistCyc}(\alpha^*)}{N^{s-1}}\right)$. After accounting for these symmetries, the rest of the random variables are all independent.

We can now add all the Bernoulli random variables corresponding to all the balls b_i that result in the same $\mathsf{DistCyc}(\alpha^*)$ (which from Lemma D.3 also means the same $\mathsf{SymCyc}(\alpha^*)$) to get a binomial distribution with the same probability. Consequently, the distribution of $\mathbb{U}_\alpha$ can be given by:

$$\mathbb{U}_\alpha \sim \sum_\tau \tau \cdot \mathsf{Binom}(\lambda_{\mathbf{u},\tau,\alpha}, \frac{s/\tau}{N^{s-1}}).$$

Notice that this also cleanly captures the distribution resultant from the two-additive sharing. Now, we can compute the expectation and variance as:

$$\mathbb{E}[\mathbb{U}_\alpha] = \sum_\tau \frac{s\lambda_{\mathbf{u},\tau,\alpha}}{N^{s-1}} = \frac{s\delta}{N^{s-1}}$$

$$\begin{aligned}\mathrm{Var}[\mathbb{U}_\alpha] &= \sum_\tau \tau^2 \cdot \lambda_{\mathbf{u},\tau,\alpha} \cdot \frac{s}{\tau N^{s-1}} \cdot \left(1 - \frac{s}{\tau N^{s-1}}\right) \\ &\leq \frac{s}{N^{s-1}} \sum_\tau \tau\lambda_{\mathbf{u},\tau,\alpha} \cdot 1 \\ &\leq \frac{s}{N^{s-1}} \cdot s\delta = \frac{s^2\delta}{N^{s-1}}\end{aligned}$$

since $\sum_\tau \tau\lambda_{\mathbf{u},\tau,\alpha}$ is maximized when $\lambda_{\mathbf{u},s,\alpha} = \delta$ and the other $\lambda_{\mathbf{u},\tau\neq s,\alpha} = 0$.
Similarly, for any other another (δ, N)-valid $\mathbf{v}$, we can compute:

$$\mathbb{E}[\mathbb{V}_\alpha] = \frac{s\delta}{N^{s-1}} \qquad \text{and} \qquad \mathrm{Var}[\mathbb{V}_\alpha] \leq \frac{s^2\delta}{N^{s-1}}$$

Now applying the Jensen's inequality $\mathbb{E}[Z] \leq \sqrt{\mathbb{E}[Z^2]}$ to the random variable $|\mathbb{U}_\alpha - \mathbb{V}_\alpha|$, we get:

$$\begin{aligned}\mathbb{E}[|\mathbb{U}_\alpha - \mathbb{V}_\alpha|] &\leq \sqrt{\mathbb{E}[(\mathbb{U}_\alpha - \mathbb{V}_\alpha)^2]} = \sqrt{\mathbb{E}[(\mathbb{U}_\alpha)^2] + \mathbb{E}[(\mathbb{V}_\alpha)^2] - 2\cdot\mathbb{E}[\mathbb{U}_\alpha]\cdot\mathbb{E}[\mathbb{U}_\alpha]} \\ &\leq \sqrt{\left(\frac{s\delta}{N^{s-1}}\right)^2 + \frac{s^2\delta}{N^{s-1}} + \left(\frac{s\delta}{N^{s-1}}\right)^2 + \frac{s^2\delta}{N^{s-1}} - \frac{2s^2\delta^2}{N^{2s-2}}} \\ &= \sqrt{\frac{2s^2\delta}{N^{s-1}}} = \frac{\sqrt{2}s}{N^{(s-1)/2}} \cdot \sqrt{\delta}.\end{aligned}$$

Therefore, we can now compute the edit distance as follows:

$$\begin{aligned}\mathsf{ED}(s\text{-}\mathsf{CNF}\text{-}\mathsf{Share}_{\mathbf{u}}, s\text{-}\mathsf{CNF}\text{-}\mathsf{Share}_{\mathbf{v}}) &= \frac{1}{2}\sum_\alpha \mathbb{E}[|\mathbb{U}_\alpha - \mathbb{V}_\alpha|] \\ &\leq \frac{N^{s-1}}{2} \cdot \frac{\sqrt{2}s}{N^{(s-1)/2}} \cdot \sqrt{\delta} \\ &\leq sN^{(s-1)/2}\sqrt{\delta}.\end{aligned}$$

□

Combining this with the result from Lemma 6.2, we can now compute the expected edit distance when $\mathbf{u}$ and $\mathbf{v}$ follow a distribution instead of being fixed.

$$\begin{aligned}\mathbb{E}_{\mathbf{u}\sim\mathcal{B},\mathbf{v}\sim\mathcal{B}'}\left[\mathsf{ED}(s\text{-}\mathsf{CNF}\text{-}\mathsf{Share}_{\mathbf{u}\ominus\mathbf{v}}, s\text{-}\mathsf{CNF}\text{-}\mathsf{Share}_{\mathbf{v}\ominus\mathbf{u}})\right] &\leq \sqrt{2N}\cdot\mathbb{E}_{\mathbf{u}\sim\mathcal{B},\mathbf{v}\sim\mathcal{B}'}\left[\sqrt{\mathsf{ED}(\mathbf{u},\mathbf{v})}\right]\\ &\leq sN^{(s-1)/2}\left(\frac{BN}{2}\right)^{1/4}\\ &\leq sN^{(2s-1)/4}B^{1/4}.\end{aligned}$$

where the second step is by the concave Jensen's inequality. Now, using Markov's inequality, we get

$$\Pr_{\mathbf{u}\sim\mathcal{B},\mathbf{v}\sim\mathcal{B}'}[\mathsf{ED}(s\text{-}\mathsf{CNF}\text{-}\mathsf{Share}_{\mathbf{u}\ominus\mathbf{v}}, s\text{-}\mathsf{CNF}\text{-}\mathsf{Share}_{\mathbf{v}\ominus\mathbf{u}}) \geq \sqrt{s}N^{(2s-3)/8}B^{1/4+\tau}] \leq \frac{\sqrt{s}\cdot N^{(2s+1)/8}}{B^{1/4+\tau}}.$$

Define this probability as ρ'_τ. Taking the total number of extra balls $\ell = B$, and $\tau = 1/8$,

$$\begin{aligned}\mathsf{SD}(s\text{-}\mathsf{CNF}\text{-}\mathsf{Share}^{\ell}_{\mathbf{u}}, s\text{-}\mathsf{CNF}\text{-}\mathsf{Share}^{\ell}_{\mathbf{v}}) &\leq \mathsf{SD}(s\text{-}\mathsf{CNF}\text{-}\mathsf{Share}^{\ell}_{\mathbf{u}\ominus\mathbf{v}}, s\text{-}\mathsf{CNF}\text{-}\mathsf{Share}^{\ell}_{\mathbf{v}\ominus\mathbf{u}})\\ &\leq (1-\rho'_\tau)\cdot S_\ell(\sqrt{s}N^{(2s-3)/8}B^{1/4+\tau}) + \rho'_\tau\cdot S_\ell(sB)\\ &\leq 1\cdot\sqrt{s}N^{(2s-3)/8}B^{1/4+\tau}\cdot\sqrt{\frac{N}{\ell}} + \frac{\sqrt{s}\cdot N^{(2s+1)/8}}{B^{1/4+\tau}}\cdot 1\\ &= \frac{\sqrt{s}\cdot N^{(2s+1)/8}}{B^{1/4-\tau}} + \frac{\sqrt{s}\cdot N^{(2s+1)/8}}{B^{\tau}} \leq \frac{2\sqrt{s}\cdot N^{(2s+1)/8}}{B^{1/8}}.\end{aligned}$$

E Proof for the Concrete Construction (Theorem 6.4)

E.1 Proof Details

Before we give the full proof, we first prove a small lemma below, which provides a way to bound the size of sub-query space in 2-additive PIR within polylogarithmic overhead of the database size.

Lemma E.1. *For any $n \in \mathbb{N}$ and $n \geq 4$, there always exists a constant c^* such that*

$$\binom{\log n + c^*\log\log n + 1}{(\log n + c^*\log\log n + 1)/2} \geq n.$$

Proof. By Stirling formula, we have

$$\binom{\log n + c^*\log\log n + 1}{(\log n + c^*\log\log n + 1)/2} \geq \frac{2\sqrt{2\pi}}{e^2}\cdot\frac{2^{\log n + c^*\log\log n + 1}}{\sqrt{\log n + c^*\log\log n + 1}}.$$

To ensure the equation in the lemma holds, it is sufficient to ensure

$$\frac{n\cdot(\log n)^c}{\sqrt{\log n + c^*\log\log n + 1}} \geq n.$$

Following above, it is sufficient to ensure

$$(\log n)^{c^*} > 3\log n,$$

and we know this is equivalent to $c^* > \frac{1}{2}\cdot(1+\frac{\log 3}{\log\log n})$. Assume $n \geq 4$, then such constant c^* exists. □

Cost analysis. Following the parameter choice specified in Section 6.3, we already have the set of parameters for OPIR and IPIR that compile. The only thing left is to choose $k, m, d, t, |\mathbb{F}|$ for OPIR (Reed-Muller PIR) and s, m', d' for IPIR (CNF PIR) based on a given constant γ.

IPIR *database size.* First, let $m = 2/\gamma$, then k and $|\mathbb{F}|$ are both $O(n^{\gamma/2})$. According to parameter choice specified in Section 6.3, the IPIR database x' (before preprocessing) has the number of entries $\Theta(n)$.

IPIR *preprocessing.* Choose $s = 2/\gamma$. Each answer in IPIR consists of $O(n^{1/s})$ monomials with coefficients represented by $\log|\mathbb{F}|$ bits, so it has the number of bits $O(n^{\gamma/2}\log n)$.

Now we want to bound the number of entries in the lookup table in the IPIR preprocessing. Let c^* be a constant; we choose $m' = \log n' + c^* \log\log n' + 1$ and $d' = m'/2 = (\log n' + c^* \log\log n' + 1)/2$. By Lemma E.1, the choice of m' results in $|\mathcal{Q}_{\mathsf{IPIR}}| = 2^{m'} = \widetilde{O}(n)$; this is also the entries in the lookup table. Plug in the answer size of IPIR above, the total number of preprocessing bits (i.e., the server storage) is $\widetilde{O}(n^{1+\gamma/2})$.

Communication and computation. For each query, the client sends $k\cdot(s+1)$ messages, each message of $\Theta(\log n)$ bits. Therefore the query size of ShPIR is $O(n^{\gamma/2}\log n)$. The answer to a query consists of $k\cdot(s+1)\cdot s$ messages, each message of $O(n^{\gamma/2}\log n)$ bits; so the answer size of ShPIR is $O(n^{\gamma}\log n)$. Since answering each query is just a table lookup, the number of bits that the server needs to read (computation cost) is exactly the same as the answer size.

A final complication is that we can get rid of the $\log n$ term by choosing a constant $\frac{2}{2/\gamma-\gamma/2+1} < \gamma' < \gamma$, and then set all parameters as above using γ' instead of γ. This results in per-query communication and computation both $O(n^{\gamma})$, and the server storage is $O(n^{\gamma'/2+2/\gamma'-1})$, which is bounded by $O(n^{2/\gamma})$ since $\gamma'/2 + 2/\gamma' - 1 < \gamma/2 + 2/\gamma' - 1 < 2/\gamma$ given the restrictions on γ' as above.

Total number of queries for security. We use Theorem 6.3 and plug in parameters for OPIR. Select parameters as described in Section 6.3, and let $m = \gamma/2$ and $s = 2/\gamma$ as above, then the $Q = |\mathcal{Q}_{\mathsf{OPIR}}| = c_1 \cdot n$, and $k = c_2 \cdot n^{\gamma/2}$ for where c_1, c_2 are constant. Using Theorem 6.3, we have $Q^{2s+1}/k\epsilon^8 = (c_1^{2s+1}/c_2^{\gamma/2}) \cdot n^{4/\gamma-\gamma/2+1}/\epsilon^8$. Let $c_0 = (c_1^{4/\gamma+1}/c_2^{\gamma/2})$, we have proved that for all $C \geq c_0 n^{4/\gamma+1}/\epsilon^8$, the composed construction has security ϵ.

The final complication is that we need to choose $\gamma' < \gamma$ in terms of efficiency (as we did for the communication cost above), therefore the resulting term is $c \cdot n^{4/\gamma'-\gamma'/2+1}$. This is asymptotically smaller than $n^{4/\gamma}$ if we choose $\gamma' < \gamma$ such that $4/\gamma' - \gamma'/2 < 4/\gamma$. Note that the restriction on γ' is equivalent to $g(\gamma') = \gamma \cdot \gamma'^2 + 8\gamma' - 8\gamma > 0$, and such γ' exists because $g(\gamma) = \gamma^3 > 0$ and g is continuous.

F Deferred Material for Lower Bound (Section 6.5)

To understand the sub-query distribution of ShPIR, we first consider a simplified case: construct a distinguisher for the OPIR sub-queries (i.e., before it splits to shares in IPIR). Then we use similar ideas to show that there exists $1/\mathsf{poly}(n)$-advantage distinguisher for the actual messages that the server observes.

F.1 Distinguishing the Queries in OPIR

As we discussed in Section 4, the Query and Answer algorithms in PIR has similar patterns as secret sharing schemes. In other words, the client "shares" its query index to sub-queries, and then combine the answers together to reconstruct the target entry. The distribution of shuffled sub-queries can be viewed as the shuffled shares of clients inputs, and our goal is to show there exists a distinguisher with inverse polynomial advantage such that it can tell between shares (sub-queries) from two sets of inputs (queried indices). Such observation allows us to adopt existing results on split and mix, which we specify below.

We borrows an idea from Ghazi et al. [GMPV20] to construct a distinguisher. In their setting, there are C clients and each client has an input $y_i \in \mathbb{Z}_p$ and is additively split into k shares in $\mathbb{Z}_p$. Then the total kC shares are randomly permuted; denote the permutation as $\pi : [kC] \to [kC]$. They construct a distinguisher $\mathcal{A}$ for the following two input cases: $(0, 0, \ldots, 0)$ and $(1, 1, \ldots, -(C-1))$, i.e., all the clients have input 0, v.s., all the clients except the last one have input 1, and the last one is set to $-(C-1)$ to ensure the sum of the inputs between the two cases are the same (the sums in both cases are 0). The distinguisher is simple: $\mathcal{A}$ accepts if $y_{\pi(1)} + y_{\pi(2)} + \ldots + y_{\pi(k)} = 0$.

The analysis for the advantage of $\mathcal{A}$ is as follows. If the above k shares do not come from the same client, then the sum is random over $\mathbb{Z}_p$; this means in both cases $\mathcal{A}$ accepts with probability $1/p$. If the above k shares come from the same client (but not the last client), then in the former case $\mathcal{A}$ accepts with probability 1 while in the latter case, $\mathcal{A}$ accepts with probability 0. Therefore, the advantage is at least $\frac{(C-1)}{\binom{kC}{k}}$, where the numerator means the k selected shares can come from the each of the C clients except the last one, and the denominator means all possible choices of k shares from the total kC shares. When the number of shares k is constant, the advantage is $1/\mathsf{poly}(C)$. The same idea can be applied for constructing a distinguisher for OPIR sub-queries.

Lemma F.1. *Given any size-n database and any k-server PIR protocol Φ such that the number of all possible k-tuples of sub-queries is K_Φ. If $K_\Phi = O(n^{t_1})$ and there are $C = O(n^{t_2})$ input indices (for some constant t_1, t_2), then there exists two input configurations in $[n]^C$, such that the statistical distance between the sub-queries generated from the two configurations is $\Omega(1/n^{t_1 t_2})$.*

Proof. We can apply similar approach (checking if randomly selected k shares sum up to zero) here, but there are some subtleties in analyzing the advantage.

Consider a PIR query algorithm that corresponds to some t-out-of-k threshold secret sharing scheme. As before, the distinguisher selects k shares; but now it may not be able to check whether the k shares reconstruct to a certain index—the selected k shares (sub-queries) may not define a valid secret (index). This complicates our analysis on distingusher advantage: when the k shares come from the same client, then the argument above remains true; but when they do not come from the same client, it is not obvious that the distinguisher accepts with the same probability in both cases. This is because the the distinguisher can exclude the invalid tuples, and it is hard to analyze the number of invalid tuples given a specific input configuration.

To tackle this, we borrow a result from Ghazi et al. [GMPV20]: for any randomized encoder (i.e., including secret sharing) that maps an input to k values, there exists a smallest integer $t \leq k$ such that, for any t shares that are not from the same client, the accepting probability is (almost) the same in the two cases; while for any t shares from the same client, the accepting probability significantly differs in the two cases. Intuitively, in our context, such t is simply the collusion threshold of PIR. Now given this result, we can let the distinguisher randomly select t sub-queries and checks if they can be reconstructed to 0 or 1.

We consider the sub-queries of PIR generated from the following two cases.

- Case 0: every client queries index 1.
- Case 1: every client queries index $i \neq 1$.

The distinguisher. Let $\mathcal{A}$ be the distinguisher for the above two cases. From the Theorem 7 in Ghaiz et.al [GMPV20], there exists a constant $t \leq t_1$, such that, given t randomly selected sub-queries, when they come from the same client, the probability that $\mathcal{A}$ accepts differ by almost 1 in the two cases; when they do not come from the same client, the probability that $\mathcal{A}$ accepts are the same.

Now we analyze the advantage of $\mathcal{A}$. The probability that the t shares come from the same client is $\frac{C}{\binom{kC}{t}}$, and when $C = O(n^{t_2})$ and $k = o(n)$, the advantage is $\Omega(\frac{1}{n^{t_1 t_2}})$.

□

Remark 12. Note that the above claim may not work when the client keeps secret states. Meanwhile, in the Reed-Muller code PIR, the client keeps the x-coordinate $r_1, \ldots, r_k$ locally, and only sends the evaluation on the x-coordinates, namely $R(r_1), \ldots, R(r_\ell)$, to the server (see details in Section 4.1.2). To simplify the proof, we can let the client send those x-coordinates as well, and this does not affect the security, since the points $(r_1, R(r_1)), \ldots, (r_k, R(r_k))$ are still pairwise independent. For completeness, for our main construction particularly (Reed-Muller code as OPIR and CNF-share PIR as IPIR), we shows that when x-coordinates are not sent to the server, there still exists an inverse polynomial lower bound. This follows from the fact that the distinguisher can correctly guess what the evaluation locations are with probability $1/|\mathbb{F}|^t$, where the denominator is polynomial as long as t is constant and $|\mathbb{F}|$ is polynomial.

F.2 Distinguishing the Shares in the Composed PIR

Consider the distinguisher for the actually sub-queries in ShPIR, it sees instead of OPIR sub-queries, the shares generated from the OPIR sub-queries, along with random noise. We first discuss below how to tackle the two issues.

For simplicity, we consider the IPIR to be the 2-additive PIR. Similar ideas also apply when IPIR is a CNF PIR; we provide details in Appendix F.3.

Handling 2-additive shares. We first give analysis when there is no noise added. Same as before, we consider two cases, where in "1-case", all clients query for index 1; and in "2-case", all the clients except the last one queries index 2, and the last client set the index to a specific i such that the sum of the shares equals the first case.

The construction is simple: $\mathcal{A}$ takes $2t$ shares, group them by 2, which results in t OPIR sub-queries; then $\mathcal{A}$ checks if the t sub-queries reconstruct to 1 or 2. If the resulting $2t$ shares are from the same client and they are paired up correctly, then the probability of $\mathcal{A}$ accepting is significantly different between the two cases. Otherwise, $\mathcal{A}$ accepts with roughly equal probability in both cases (since the reconstruction will give a random index).

Handling the noise. Now we construct the distinguisher when dummy queries are added. Note that adding dummy queries is equivalent to having more clients with random inputs. Consider adding C more clients in the above example for additive shares. When the t shares are not from the same party, $\mathcal{A}$ accepts with the same probability in both cases. When the t shares come from the same party, $\mathcal{A}$ accepts with probability roughly $1/2$ in the former case, but only with probability roughly 0 in the latter case. Here the advantage is not close to 1, the advantage is simply decreases by a constant factor of c if each clients adds the a constant c number of noise.

F.3 Proof of Theorem 6.5

From Lemma F.1, there exists a constant t such that by picking t shares, one can distinguish between two certain sets of query indices.

Now we utilize such t to construct a distinguisher with $1/\mathsf{poly}(n)$ advantage for the following two cases:

- Case 0. All clients except the last one query index 1; the last one queries a random $i \in [n]$.
- Case 1. All clients except the last one query index 2, the last one queries a random $i' \in [n]$.

The distinguisher. First, $\mathcal{A}$ picks $2t$ random shares, group them by two. If the resulting grouping does not result in valid encoding in IPIR, then $\mathcal{A}$ outputs reject and terminate; otherwise $\mathcal{A}$ continues, and the grouped shares define the "sub-queries" of OPIR. Then, $\mathcal{A}$ reconstructs the t "sub-queries" and outputs accept if they reconstruct to 1. Otherwise, it outputs reject.

Analysis of advantage. When the $2t$ shares are from the same client (except the last one) and they are grouped correctly, $\mathcal{A}$ accepts with probability at least $1/2$ in Case 0 (assuming one noise per client); and it accepts with probability $\frac{1}{2|\mathcal{Q}_{\mathsf{OPIR}}|}$ in Case 1. When the $2t$ shares are not from the same client, if $\mathcal{A}$ does not have a valid grouping, it will reject in both cases, and if $\mathcal{A}$ does have a valid grouping, then it will accept with probability $\frac{1}{|\mathcal{Q}_{\mathsf{OPIR}}|}$ in both cases.

Putting the above together, the advantage is at least $\frac{1}{2} \cdot \frac{(C-1)\binom{k}{t}}{\binom{kC}{2t}}$; since $C = O(n^{t_2})$, $k = O(n^{1/m})$ and t, k, t_2 are all constant, then the advantage is asymptotically at least $1/n^{t_1 t_2}$.

Remark 13 (Lower bound on security for CNF IPIR). The above lower bound also applies for CNF share; after all, CNF shares provides less security. A tweak to the adversary above will directly result in a distingusher for s-CNF-ShPIR: for each share, $\mathcal{A}$ only takes the first component as the share.