

Towards Doubly Efficient Private Information Retrieval*

Ran Canetti[†]
BU & TAU

Justin Holmgren[‡]
MIT

Silas Richelson[§]
UC Riverside

September 25, 2017

Abstract

Private Information Retrieval (PIR) allows a client to obtain data from a public database without disclosing the locations accessed. Traditionally, the stress is on preserving sublinear work for the client, while the server’s work is taken to inevitably be at least linear in the database size. Beimel, Ishai and Malkin (JoC 2004) show PIR schemes where, following a linear-work preprocessing stage, the server’s work per query is sublinear in the database size. However, that work only addresses the case of multiple non-colluding servers; the existence of single-server PIR with sublinear server work remained unaddressed.

We consider single-server PIR schemes where, following a preprocessing stage in which the server obtains an encoded version of the database and the client obtains a short key, the per-query work of both server and client is polylogarithmic in the database size. Concentrating on the case where the client’s key is secret, we show:

- A scheme, based on one-way functions, that works for a bounded number of queries, and where the server storage is linear in the number of queries plus the database size.
- A family of schemes for an unbounded number of queries, whose security follows from a corresponding family of new hardness assumption that are related to the hardness of solving a system of noisy linear equations.

We also show the insufficiency of a natural approach for obtaining doubly efficient PIR in the setting where the preprocessing is public.

1 Introduction

Enabling clients to query remote databases while preserving privacy of the queries is a basic challenge in cryptography. With the proliferation of huge databases stored and managed by powerful third parties, this challenge becomes ever more relevant.

One of the more basic formulations of this multi-faceted problem is the concept of *Private Information Retrieval (PIR)* [CKGS98]. Here the client is interested in learning the contents of specific addresses in the database, while preventing the server controlling the database (or, simply, the database) from learning these addresses. The goal here is to minimize communication and work for the client.

There are two general types of PIR schemes: Multi-server schemes whose security relies on the assumption that servers do not collude, but are otherwise information theoretic (see e.g. [CKGS98, BI01]), and single-server schemes which are based on computational assumptions, often of a structured nature (e.g., [KO97, CMS99]). Still, in both cases, the per-query work of the server is traditionally taken to be at least linear in the database size — else the server can “obviously” somewhat localize the requested address. Indeed, this linear server overhead is a main bottleneck for deployment (see e.g. [CSP⁺16]).

Is this bottleneck really inevitable? A first indication that this might not be the case is the body of work on oblivious RAM [GO96, Ajt10, DMN11, SCSL11]: Here a client can indeed access a database in a privacy preserving

*Supported by the NSF MACS project.

[†]Member of CPIIS. Supported in addition by ISF Grant 1523/14.

[‡]Supported by NSF MACS - CNS-1413920, DARPA IBM - W911NF-15-C-0236, and SIMONS Investigator award Agreement Dated 6-5-12

[§]Work done while at MIT and Boston University.

manner, with polylogarithmic overhead for both client and database (following an initial poly-time preprocessing stage). However, oblivious RAM schemes inherently require the client to (a) keep secret state and (b) be able to update the database. This is so even if the client only wants to read from the database. Furthermore, if the database is not trusted for correctness then the client needs to be able to continually update its local state. Consequently, a database cannot serve multiple clients without having the clients continually coordinate with each other.

An indication that these restrictions might not be necessary either is the work of Beimel, Ishai and Malkin [BIM04]. They present PIR schemes where, following an initial polynomial-time preprocessing stage, the client and servers each are stateless and incur a per-query overhead that is sublinear in the database size. However, that work considers only the multi-server setting, and furthermore the number of servers is tied to the “level of sublinearity” of the scheme. This leaves open the following question:

Can we construct a single-server PIR scheme where the client has no updatable state and where the per-query work of both the client and the server is sublinear in the database size, potentially with a more expensive preprocessing stage?

Paraphrasing [GR17], we call this primitive doubly efficient PIR (DEPIR).

1.1 Our contributions

We provide some positive answers to this question, along with some impossibility results and cryptanalysis of our candidate schemes. Our DEPIR schemes start with an initial preprocessing stage which takes the database as input and hands a preprocessed version of the database to the server and (short) key to the client. We distinguish between the *public preprocessing* case, where the random choices made during preprocessing are public, the *public client* case, where the preprocessing may use secret randomness but the client’s long-term key is public, and the *designated client* case where the client’s key remains hidden from the server. In all cases the client maintains no state across queries other than the long-term key, and the database is read-only.

We have no positive results for the first two cases. In fact, we demonstrate that a natural and general approach towards a public preprocessing solution is doomed to fail. We leave progress on this fascinating question to future research. We then concentrate on the designated client case. Here we show:

1. A designated-client scheme for a bounded number of queries, whose security can be based on any one way function. Given a bound B on the number of client queries, the size of the preprocessed database is $\tilde{O}(B + \text{poly}(N))$, where N is the database size. The client keeps a short secret key of size λ , the security parameter. The per-query client and server overheads are $\lambda \cdot \text{polylog}(N, B)$ and $\text{polylog}(N, B)$, respectively. (We compare the performance of this scheme to that of the trivial stateful scheme where in preprocessing the client hands the server B independently permuted copies of the database, and then uses a different copy for each query. Here the online work is only $\log N$, but the size of the preprocessed database is BN . This means that the amortized space overhead in this scheme is N , whereas in our scheme it is $\text{polylog}(N)$.) We then demonstrate the tightness of the analysis in two ways:
 - (a) We demonstrate an efficient attack on this scheme which is applicable as soon as the number of queries exceeds the designated bound B .
 - (b) We demonstrate the failure of a natural approach to extending the scheme while preserving its proof structure¹. Along the way, we also show a tight quantitative extension of the impossibility result from [CKGS98] regarding the communication complexity in standard single-server PIR, and a tight bound on the size of the key in an information-theoretic designated-client PIR with preprocessing.
2. A candidate extension of the scheme from (1) to the case of unbounded queries. The extended scheme provides a natural tradeoff between complexity and potential security, where in the best case the complexity parameters are comparable to those of the bounded scheme with $B = 1$. We are unable to prove security of this scheme based on known assumptions. Instead, we introduce a new computational problem, which we call the **hidden permutation with noise (HPN)** problem, and reduce the security of our scheme to the assumption that HPN

¹Specifically, using pseudo-random functions and pseudo-random permutations to compress a long key for a statistically secure scheme.

is hard. HPN is a noisy learning problem and is thus superficially similar to other “standard” cryptographic assumptions (e.g. LWE, LPN), but we do not know of any reduction to these assumptions.

3. We take first steps towards analyzing the hardness of HPN. Security of the candidate scheme from (2) can be rephrased as the hardness of the adaptive, decision version of HPN. We then prove that the adaptive decision version is no harder than the static (selective), search version. This allows future investigation of the hardness of HPN to focus on the latter version, which is structurally simpler and more basic.
4. We also consider a number of other methods for extending the scheme from (1) to the case of unbounded queries. However, while these other methods may well provide additional security on top of the HPN-based scheme described above, we are unable to make significant headways either towards cryptanalysis of these methods, or towards reducing security to a simpler problem.

In the rest of the introduction we describe our contributions in more detail.

Defining PIR security with preprocessing. We formulate the security of PIR schemes with preprocessing in a range of settings, including the ones mentioned above and some variants. A PIR scheme with preprocessing consists of five algorithms (Keygen, Process, Query, Resp, Dec), as follows. Keygen takes the security parameter λ and samples a key K . Process takes K and a database $DB \in \Sigma^N$ for some alphabet Σ and samples a preprocessed database \widetilde{DB} to be handed to the server. Query takes K and an address $i \in \{1, \dots, N\}$, and samples a query q and local state s . Resp computes the server’s response d given q and RAM access to \widetilde{DB} , and Dec outputs the decoded value given K , s , and d . Giving Resp oracle (random-)access to \widetilde{DB} enables it to be sublinear in N . It is stressed that the client is stateless, apart from the long-term key K and the short-term state s between sending a query and obtaining its response.

The correctness requirement is obvious. Double efficiency means that Query, Resp, and Dec each run in time $o(N) \cdot \text{poly}(\lambda)$ (ideally, they should be polylogarithmic in N), and additionally Process runs in time $\text{poly}(N, \lambda)$. For security, we consider three main cases (plus several variants).

In the **designated client** case, security is defined by requiring that for any polytime stateful adversary \mathcal{A} there exists a polytime stateful simulator \mathcal{S} such that \mathcal{A} can distinguish between the following two games only with negligible probability: In the real game, \mathcal{A} chooses a database DB , obtains a preprocessed database $\widetilde{DB} \leftarrow \text{Process}(K, DB)$ where $K \leftarrow \text{Keygen}(\lambda)$, and then repeatedly and adaptively generates an index $i \in [N]$ and obtains $\text{Query}(K, i)$. In the ideal game, \mathcal{A} chooses a database DB , obtains a simulated preprocessed database $\mathcal{S}(DB)$, and then repeatedly and adaptively generates an index $i \in [N]$ and obtains $\mathcal{S}()$. (If \mathcal{S} gets only N rather than DB then we say that the scheme guarantees also **database privacy**.)

In the **public client** case, the above real game is modified so that \mathcal{A} obtains $K \leftarrow \text{Keygen}(\lambda)$ before generating DB ; in the ideal game K is generated by \mathcal{S} . In the **public preprocessing** case we assume, in addition, that Keygen simply outputs its random input, and Process is deterministic. (Other variants are considered within.)

We remark that our definitional style is inspired by UC security. Indeed, our game-based definitions can be reformulated as realizing the appropriate “ideal PIR functionalities”.

Candidate constructions and analysis. For simplicity, we restrict attention to DEPIR schemes where the client’s query consists of a list of addresses in the encoded database \widetilde{D} , and the server answers with the contents of \widetilde{D} in these addresses. (Since we are shooting for polylogarithmic communication complexity, not much generality is lost by this simplification.)

When viewed this way, the encoding of a database bears resemblance to locally decodable codes: The i^{th} symbol of D should be decodable by accessing only few locations of the encoded version \widetilde{D} . Here however we have the added requirement that the locations queried should be simulatable without knowing the original addresses i . On the other hand, we do not have any error-correction requirements.

A natural approach, which we follow, is thus to start from an existing locally decodable code and try to modify it to obtain privacy. Specifically, we start from the following variant of Reed-Muller codes: In order to encode a database $D \in \{0, 1\}^N$ choose a field \mathbb{F} of size $\text{polylog}(N)$ and a subset $H \subset \mathbb{F}$ of size $\log N$. The database is viewed as a function $D : H^m \rightarrow \{0, 1\}$, where $m = \log N / \log \log N$ so that $|H^m| = N$. With this setup in place, \widetilde{D} is the truth table of the *low degree extension* of D . That is, $\widetilde{D} = \{\widehat{D}(x) : x \in \mathbb{F}^m\}$, where $\widehat{D} : \mathbb{F}^m \rightarrow \mathbb{F}$ is the unique m -variate

polynomial of degree at most $(|H| - 1)$ in each variable, such that $\hat{D}(i) = D(i)$ for all $i \in H^m$. The total degree of \hat{D} is $d = O(\log^2 N)$. To query the database at $i \in H^m$ (i.e., to decode $D(i)$ from \tilde{D}), the client chooses a line $\varphi : \mathbb{F} \rightarrow \mathbb{F}^m$ such that $\varphi(0) = i$, and sets $q = (\varphi(1), \dots, \varphi(k))$ where $k = d + 1$. Upon receiving the server's response $(\mathbf{x}_1, \dots, \mathbf{x}_k)$, client recovers $D(i) = \varphi(0)$ by interpolation (note that $(\mathbf{x}_1, \dots, \mathbf{x}_k)$ all lie on a curve of degree at most d).

The above scheme is clearly insecure as the server can easily interpolate $i = \varphi(0)$ from the client's query $(\varphi(1), \dots, \varphi(k))$. In this work we study three natural and orthogonal alterations to this basic scheme which attempt to prevent the server from interpolating, while still allowing the client to do so. As a preliminary step to all alterations, we let φ be a random degree- d' polynomial, where $d' = \text{polylog}(N)$, rather than a random line. (Still, we let $D(i) = \varphi(0)$.) This means that, in order to allow the client to interpolate $\tilde{D}(\varphi(0))$, we should set $k \geq dd' + 1$. As we'll see, setting d' to be super-constant will be important for the security of our schemes in various ways. The three alterations are:

1. The client uses *secret* evaluation points $(\alpha_1, \dots, \alpha_k)$ rather than $(1, \dots, k)$. The evaluation points can be chosen randomly once and remain fixed throughout, or alternatively chosen anew for each query.
2. The client introduces noise by adding some random points in \mathbb{F}^m to the query, at random locations. (There are a number of different variants here, depending on the noise structure.)
3. At preprocessing, the client first encrypts all elements of the database using symmetric encryption. Next it encodes the database to obtain \tilde{D} . Finally, it secretly and pseudorandomly permutes the elements of \tilde{D} before handing it to the server. The client keeps the encryption key, as well as the key of the pseudorandom permutation.

That is, let $\pi \in \text{Perm}(\mathbb{F}^m)$ be a pseudorandom permutation. (Since the domain is polynomial in size, use e.g. [MRS09].) The precomputed database \tilde{D} is now the truth table of the function $\hat{D} \circ \pi^{-1} : \mathbb{F}^m \rightarrow \mathbb{F}$. To query address i , client draws φ and computes $(\mathbf{x}_1, \dots, \mathbf{x}_k)$ as before, but sends the query $(\mathbf{x}'_1, \dots, \mathbf{x}'_k)$ where $\mathbf{x}'_i = \pi(\mathbf{x}_i)$. The client uses the responses (a_1, \dots, a_k) to interpolate the encrypted $D(i)$, and then decrypts to obtain the actual value.

The various combinations of these three ideas suggest several possible DEPIR schemes. Note that a scheme based on either of the first two ideas (or both) but not the third would be public preprocessing. The third idea results in a designated client scheme. We briefly review how our main results are mapped to these three ideas.

No Public-Client DEPIR via Linear Codes: In Section 3, we prove that any combination of the first two ideas alone is insecure. More generally, we show that any scheme where the preprocessed database \tilde{D} is obtained just by encoding D via some explicit linear code cannot be secure. This holds regardless of which query and response mechanism is used.

Bounded Security, Unbounded Insecurity via (3): In Section 4, we show that alteration (3) by itself suffices to obtain our bounded-query result stated above. In fact, we show that security holds even if, instead of starting from Reed-Muller codes, we start from any locally decodable code in a rather general class of codes.

On the other hand, we demonstrate an explicit attack on the scheme, for the case of Reed-Muller codes, if the client asks even slightly more queries than the bound allows.

Candidate Schemes via other combinations: We observe that the previous attack is thwarted by using either (1) and (3) together, or using (2) and (3) together. In fact, we were unable to break either one of these two candidates, with any non-trivial level of noise in (2). We thus suggest them as target for further cryptanalysis. In fact it is reasonable to also propose the scheme that combines all three ideas.

We note however that, while it is tempting to assume that adding alterations (e.g., adding noise or moving from fixed evaluation points to hidden or random evaluation points) increases security, or at least does not harm security, we cannot always back this intuition by actual reductions. For instance we do not currently see a way to argue that, a scheme that uses all three alterations is always no less secure than a scheme that uses only alterations (2) and (3).

Security reduction: We concentrate on the following “minimal” variant of a scheme that combines ideas (2) and (3): We choose a random set T of l indices in $[k + l]$. We then run alteration (3) with $k + l$ evaluation points, obtain a query q_1, \dots, q_{k+l} , and then for each $j \in T$ we replace q_j with a random point in the domain. In Section 5, we reduce the security of this scheme to the computational hardness of a *search* problem that is roughly sketched as follows. As already mentioned, we call this the *hidden permutation with noise (HPN)* problem:

In the HPN problem, a random permutation $\pi \in \text{Perm}(\mathbb{F}^m)$ is chosen, where $|\mathbb{F}| = \text{polylog}(N)$ and $|\mathbb{F}^m| = \text{poly}(N)$. The problem is to compute π given samples chosen as follows: First a random set T of l indices out of $[k + l]$ is chosen. Next, draw $\text{poly}(N)$ samples from the following distribution $H_{\pi, T}$: Choose $z \leftarrow \mathbb{F}$, a degree- d polynomial $\varphi : \mathbb{F} \rightarrow \mathbb{F}^m$ with $\varphi(z) = 0$. Now, for $i \in T$ choose y_i randomly from \mathbb{F}^m . For $i \notin T$, let $y_i = \varphi(i)$. The sample is $(\pi(z), \pi(y_1), \dots, \pi(y_{k+l}))$. The parameters are set so that $k \approx d^2$; however l can be significantly larger, so a sample may not uniquely determine φ even if π is known.

Note that a sample from $H_{\pi, T}$ directly corresponds to a query in the scheme. In other words, security of the scheme corresponds directly to a decisional variant of HPN with adaptively chosen free coefficients for φ . In contrast, the HPN problem as formulated above is a search problem with non-adaptive input.

One may of course consider also another variant of this scheme, where the client chooses a new set T for each query. While this variant indeed appears to be harder to cryptanalyze, we are not able to argue that it is no less secure than the above, fixed- T variant. Furthermore, we were unable to extend the decision-to-search reduction to this variant.

Finally, we note that only the first alteration above (namely, using random evaluation points) is specific to Reed-Muller codes. The other two are generic and apply to any locally decodable code, opening other potential routes to DEPIR schemes. In fact our bounded-query scheme in Section 4 is stated (and proved secure) generically in terms of any locally decodable code whose decoding queries are t -wise uniform for sufficiently large t .

Related Work. There are several existing hardness assumptions about polynomials in the literature, which do not appear to be related to ours. In particular, we point out the “noisy polynomial interpolation” problem, introduced by Naor and Pinkas [NP06] and (somewhat) cryptanalyzed by Bleichenbacher and Nguyen [BN00]. Two main differences between this assumption and our HPN assumption are that (i) we completely hide the algebraic structure of the underlying field by permuting the polynomial’s domain, and (ii) we work with multivariate polynomials rather than univariate polynomials, which can sometimes make reconstruction problems much more difficult [GKS10].

Coppersmith and Sudan [CS03] show how to remove noise from codes based on multi-variate polynomials. However, their techniques do not appear to extend to our case of codes concatenated with a hidden permutation.

1.2 Independent Work

The problem we consider has been independently studied by Boyle, Ishai, Pass and Wootters [BIPW17]. The two works consider the same problem of sublinear-time PIR with preprocessing and propose similar candidate solutions based on secretly permuted Reed-Muller codes.

The notions of designated-client (resp., public-client) doubly-efficient PIR from the present work correspond to the notions of OLDC (resp., pk-PIR) from [BIPW17] (with the exception that they make the additional restriction of non-adaptive queries). We provide an overview of the main differences between the two works below.

The main contributions of [BIPW17] beyond those of this work include:

1. A general transformation from (designated-client) OLDC to (public-client) pk-PIR by applying VBB obfuscation to the query generation algorithm and an authenticated version of the decoding algorithm. This yields an explicit candidate construction of pk-PIR.
2. Two types of barriers: A “data structures barrier,” suggesting that even a very strong form of pk-PIR, with deterministic encoder and non-adaptive queries, would be difficult to unconditionally rule out; and an “LDC barrier,” showing that OLDC implies traditional LDC, effectively imposing a limitation on the space of possible candidates.

3. Ruling out (under standard assumptions) a natural “learning” approach for generically breaking constructions based on secret linear codes, by using the power of span programs.
4. A proof that any OLDC (or, DEPIR scheme) implies a one-way function.

The main contributions of this work beyond those of [BIPW17] include:

1. A different variant of the designated-client (OLDC) candidate in which the curve evaluation points used by the decoder are fixed (or made public) but some of the points on the curve are replaced by random noise. A combination of random noise with secret evaluation points is also proposed as a potentially more conservative candidate.
2. A search-to-decision reduction for a restricted case of the above fixed-evaluation-point variant, where the location of the noise elements is the same for all queries.
3. An efficient variant of the designated client scheme, that is secure in the *bounded-query* case assuming one way functions.

2 Defining Doubly Efficient PIR

We define doubly-efficient PIR (DEPIR) schemes within a number of settings. While we only construct DEPIR schemes in few of these settings, we hope that the definitions will be useful as a basis for future work. In all settings, we consider information retrieval schemes with preprocessing that consist of the following five algorithms:

- Keygen takes the security parameter 1^λ and samples a key k .
- Process takes k and a database $DB \in \Sigma^N$ and outputs a preprocessed database \widetilde{DB} .
- Query takes k and index $i \in \{1, \dots, N\}$ and outputs a query q and local state st .
- Resp takes q and \widetilde{DB} and returns the server answer a .
- Dec takes k, st, a and returns a data element in Σ .

The correctness and efficiency requirements are obvious:

Definition 1 (Correct & Doubly Efficient Information Retrieval). *An information retrieval scheme with preprocessing $\Pi = (\text{Keygen}, \text{Process}, \text{Query}, \text{Resp}, \text{Dec})$, is correct if the correctness error of Π is negligible in λ , where Π has correctness error ϵ if for all $DB \in \Sigma^N$ and $i \in [N]$ we have:*

$$\Pr [\text{Dec}(st, a) \neq DB_i] \leq \epsilon$$

in the probability space defined by sampling

$$K \leftarrow \text{Keygen}(1^\lambda), \widetilde{DB} \leftarrow \text{Process}(K, DB), (q, st) \leftarrow \text{Query}(K, i), a \leftarrow \text{Resp}^{\widetilde{DB}}(q)$$

If $\epsilon = 0$, then Π is perfectly correct.

Π is doubly efficient if all algorithms are polynomial in λ , $|\widetilde{DB}| = \text{poly}(\lambda, N)$, and Query, Resp, Dec are sublinear in N , where Resp is given RAM access to \widetilde{DB} . (Ideally, these algorithms are polylogarithmic in N .)

We consider several levels of security. In all levels, the database and the indices to be read by the client are assumed to be chosen adversarially (“by the environment”). Also, in all levels we only consider honest-but-curious database servers, i.e. we trust the database server to run Resp (and sometimes also Process) as specified. We then consider three main levels with some variants:

Designated client: Here the key k is known only to the client. We consider two variants: One where the server does not even learn the database itself, and the other where the server may learn the database but does not learn the queried locations. We also consider the case of security for only a bounded number of queries.

Public client: Here k is assumed to be public (and honestly generated); however, the randomness used to generate k is assumed to remain hidden. Since Query is stateless, this means that many clients may query the database concurrently. The server learns the database but not the queried locations. Here (and in the next variant) we consider two cases, depending on whether the database is chosen before k is known, or adaptively, depending on k . We also consider the case where Keygen generates an additional secret key that is used only by Process.

Public preprocessing: This is the same as public client, except that the randomness used by Keygen is public (or, equivalently, Keygen simply outputs is random input.)

Our formal notions of security are inspired by UC security, and can be formulated via realizing appropriate variants of an “ideal PIR functionality”. However, for self-containment we present them directly via the following games. (To simplify presentation the notation below considers adversaries and simulators that are stateful throughout the interaction.)

Definition 2 (Designated-Client & bounded-query DEPIR). A DEPIR scheme $\Pi = (\text{Keygen}, \text{Process}, \text{Query}, \text{Resp}, \text{Dec})$ is Designated Client if for any PPT \mathcal{A} there exists a polytime simulator \mathcal{S} such that $\text{REAL}_{\mathcal{A},\Pi} \approx \text{IDEAL}_{\mathcal{A},\mathcal{S}}$, where:

$\text{REAL}_{\mathcal{A},\Pi}$ is the output of \mathcal{A} in the following interaction:

1. $\mathcal{A} \rightarrow \text{DB} \in \Sigma^N$ and obtains $\widetilde{\text{DB}} \leftarrow \text{Process}(K, \text{DB})$, where $K \leftarrow \text{Keygen}(\lambda, N)$.
2. Repeat until \mathcal{A} generates final output:
 - (a) \mathcal{A} generates an index $i \in [N]$ and obtains $q \leftarrow \text{Query}(K, i)$.

$\text{IDEAL}_{\mathcal{A},\mathcal{S}}$ is defined as the output of \mathcal{A} in the following interaction:

1. \mathcal{A} outputs a database $\text{DB} \in \Sigma^N$ and obtains $\widetilde{\text{DB}} \leftarrow \mathcal{S}(\lambda, \text{DB})$;
2. Repeat until \mathcal{A} generates final output:
 - (a) \mathcal{A} generates an index $i \in [N]$ and obtains $q \leftarrow \mathcal{S}()$.

If in the above interaction \mathcal{S} obtains only N rather than the entire DB then we say that the scheme has also database privacy.

If Keygen obtains an additional input B , and the privacy requirement is modified so that \mathcal{A} can generate at most B queries, then we say that the scheme is B -bounded DEPIR. In this case we define the amortized storage overhead of the scheme to be $|\widetilde{\text{DB}}|/B$.

Definition 3 (Public-Client & Public-Preprocessing DEPIR). A DEPIR scheme $\Pi = (\text{Keygen}, \text{Process}, \text{Query}, \text{Resp}, \text{Dec})$ is Public-Client if for any polytime \mathcal{A} there exists a polytime simulator \mathcal{S} such that $\text{REAL}_{\mathcal{A},\Pi} \approx \text{IDEAL}_{\mathcal{A},\mathcal{S}}$, where:

$\text{REAL}_{\mathcal{A},\Pi}$ is the output of \mathcal{A} in the following interaction:

1. $\mathcal{A}(K) \rightarrow \text{DB} \in \Sigma^N$, where $K \leftarrow \text{Keygen}(\lambda, N)$; \mathcal{A} obtains $\widetilde{\text{DB}} \leftarrow \text{Process}(K, \text{DB})$.
2. Repeat until \mathcal{A} generates final output:
 - (a) \mathcal{A} generates an index $i \in [N]$ and obtains $q \leftarrow \text{Query}(K, i)$.

$\text{IDEAL}_{\mathcal{A},\mathcal{S}}$ is defined as the output of \mathcal{A} in the following interaction:

1. $\mathcal{A}(K) \rightarrow \text{DB} \in \Sigma^N$, where $K \leftarrow \mathcal{S}(\lambda, N)$; \mathcal{A} obtains $\widetilde{\text{DB}} \leftarrow \mathcal{S}(\text{DB})$.
2. Repeat until \mathcal{A} generates final output:

(a) \mathcal{A} generates an index $i \in [N]$ and obtains $q \leftarrow \mathcal{S}()$.

If in the above interaction Keygen outputs its random input, and Process uses no additional randomness other than K then we say that the scheme is **public preprocessing**.

If in the above interaction \mathcal{A} obtains K only after generating $\widehat{\text{DB}}$ then we say that the scheme has **non-adaptive database generation**.

If in the above interaction \mathcal{S} obtains $K = \text{Keygen}(\lambda, N)$ rather than generating it, then we say that the scheme has **global key generation**. If in the above interaction Keygen generates an additional key that is used by Process and otherwise remains unknown, then we say that the scheme is **public client with secret preprocessing key**.

3 Failure of A Natural Approach for Public-Preprocessing DEPIR

We present an approach for constructing public-preprocessing DEPIR, and demonstrate its failure. Whether public-preprocessing doubly efficient PIR schemes exist is left as a fascinating open question.

As sketched in the introduction, a natural approach to constructing public-preprocessing DEPIR is to view the database $\text{DB} \in \{0, 1\}^N$ as a function $\text{DB} : H^m \rightarrow \{0, 1\}$, where $H \subset \mathbb{F}$ with $|H| = \log N$, \mathbb{F} is a finite field of order $\text{polylog}(N)$, and $m = \frac{\log N}{\log \log N}$. The encoding of DB is done by extending it to an m -variate polynomial $\widehat{\text{DB}} : \mathbb{F}^m \rightarrow \mathbb{F}$ of degree $|H| - 1$, where $\widehat{\text{DB}}(x) = \text{DB}(x)$ for all $x \in H^m$.

One may hope to construct query distributions $\{\mathcal{Q}_i\}_{i \in H^m}$ such that

1. (**Interpolability**) When sampling $Q \leftarrow \mathcal{Q}_i$, it holds that $\widehat{\text{DB}}|_Q$ determines $\text{DB}(i)$. That is, any m -variate polynomial g of degree $|H| - 1$ which agrees with $\widehat{\text{DB}}$ on Q also agrees with $\widehat{\text{DB}}$ (and therefore DB) on i .
2. (**Privacy**) \mathcal{Q}_i computationally hides i . That is, for any $i \neq i'$, \mathcal{Q}_i and $\mathcal{Q}_{i'}$ are computationally indistinguishable.

Such a construction would immediately give a (public-client) PIR scheme. The server just stores the truth table of $\widehat{\text{DB}}$, and the client makes queries to i by sampling $Q \leftarrow \mathcal{Q}_i$, asking the server for $\widehat{\text{DB}}|_Q$, and interpolating the returned values to obtain $\text{DB}(i)$.

For example, one may suggest to construct \mathcal{Q}_i by sampling a uniformly random curve $\gamma : \mathbb{F} \rightarrow \mathbb{F}^m$ of degree $t = \log^2 k$ such that $\gamma(0) = i$. That is, \mathcal{Q}_i is defined as $\{\gamma(x_i)\}_{i=0}^d$, where $\{x_i\}_{i=0}^d$ are uniformly random distinct points in $\mathbb{F} \setminus \{0\}$, and $d = t \cdot m \cdot (|H| - 1)$ is the degree of $\widehat{\text{DB}} \circ \gamma$; A natural justification here is that interpolating $\{\gamma(x_i)\}$ to find $\gamma(0)$ seems to require knowledge of the evaluation points $\{x_i\}$. One can also include in the query some number of random points in \mathbb{F}^m to make interpolation look even harder.

This template can be further generalized by replacing $\widehat{\text{DB}}$ with any locally decodable code (LDC) encoding of DB , and appropriately adapting the notion of interpolability.

However, we show that even this general template fails, as long as the LDC in use is *linear* — which rules out the vast majority of the LDCs studied in the literature. We are inspired by a recent work of Kalai and Raz [KR17], which shows (among other things) the insecurity of the Reed-Muller instantiation of this template.

Definition 4. Let $\mathcal{C} : \Sigma^N \rightarrow \Sigma^M$ be a code, and let Q be a subset of $[M]$. We say that Q determines $i \in [N]$ if for every $m, m' \in \Sigma^N$ for which $\mathcal{C}(m)|_Q = \mathcal{C}(m')|_Q$, it holds that $m_i = m'_i$.

Proposition 1. Let $\mathcal{C} : \mathbb{F}^N \rightarrow \mathbb{F}^M$ be a linear code. Then there is a $\text{poly}(M)$ -time algorithm which takes as input a set of queries $Q \subseteq [M]$, and outputs all indices $i \in [N]$ which are determined by Q . Furthermore, there are at most $|Q|$ such indices.

Proof. Let $G \in \mathbb{F}^{M \times N}$ be the generator matrix for \mathcal{C} . Then Q determines i iff the i^{th} standard basis vector $\mathbf{e}_i \in \mathbb{F}^N$ is spanned by the rows of G which are indexed by Q . This criterion is efficiently checkable by Gaussian elimination.

To see the “furthermore” part of the proposition, let A denote the set of $i \in [N]$ which are determined by Q . If $|A|$ were larger than $|Q|$, then $(Gm)_Q$ would be a compressed encoding of m_A , which is impossible since m_A was arbitrary. □ □

4 Bounded-Query Designated-Client DEPIR

We consider the case of bounded-query, designated-client DEPIR. We first present a trivial scheme with minimal online work but with large space overhead for the server which also requires a stateful client. Next we present our main bounded-query scheme, which is based on any family of locally decodable codes. When instantiated with the Reed-Muller family of codes (with a generalized decoding procedure), we obtain the following parameters, for a database of size N and query bound B :

- The prover and verifier both do $\text{polylog}(N)$ online work
- The processed database size and secret key size are both $|\widetilde{\text{DB}}| = \tilde{O}(B + \text{poly}(N))$.

We emphasize this holds with a *stateless client*. Our construction improves significantly on the trivial scheme, which only supports a single query and cannot be simply scaled up by repetition without having the client maintain updatable long-term state.

Database encryption. In both schemes the first step in the preprocessing of the database is to encrypt each entry, using some semantically secure symmetric encryption, with a key that's part of the clients secret key. Notice that this step makes the encrypted database computationally independent from the plaintext database. For simplicity of presentation, we omit the encryption step from the description of both schemes, and instead assume that the adversary does not see the plaintext database. Instead, it sees only the preprocessed database and the queries. (It is also possible to obtain statistical independence between the plaintext databases and the preprocessed one by using perfectly secure encryption such as one time pad, at the cost of a longer client secret key.)

Organization. The trivial, stateful scheme is presented in Section 4.1). The main scheme is presented and analyzed in Section 4.2. Optimality of the analysis is demonstrated in Sections 4.3 and 4.4. Section 4.3 presents an efficient attack that kicks in as soon as the number of queries exceeds $\lfloor \mathbb{F}^m \rfloor$. The attack is specific for the Reed-Muller instantiation of the scheme. Section 4.4 provides a more general bound on the key size and communication complexity of any statistically-secure designated client DEPIR.

4.1 A trivial scheme

We note that it is trivial to construct a one-round designated-client PIR scheme with perfect correctness and perfect 1-query security, with server storage $\tilde{O}(N)$ and server work $O(1)$:

- $\text{Keygen}(1^\lambda, N)$ samples a uniformly random permutation $\pi : [N] \rightarrow [N]$, and outputs $K = \pi$.

- $\text{Process}(K, \text{DB})$ outputs $\widetilde{\text{DB}}$, where

$$\widetilde{\text{DB}} : [N] \rightarrow \{0, 1\}$$

$$\widetilde{\text{DB}}(i) = \text{DB}(\pi(i)).$$

- $\text{Query}(K, i)$ outputs (q, st) , where $q = \pi(i)$ and st is the empty string.
- $\text{Resp}(\widetilde{\text{DB}}, q)$ outputs $\widetilde{\text{DB}}(q)$.
- $\text{Dec}(\text{st}, a)$ outputs a .

Extensions and Shortcomings. If the client is allowed to keep long-term state (i.e. remember how many queries it has made), then one can obtain a B -query scheme by concatenating B single-query schemes, resulting in server (and client) storage which is $\tilde{\Theta}(BN)$. With a *stateless* client however, it is not even clear how to support 2 queries. Furthermore, the storage cost of $\tilde{\Theta}(BN)$ leaves much to be desired.

4.2 A scheme based on LDCs and random permutations

We show a scheme with a stateless client, parameterized by a query bound B , which achieves B -bounded security and server storage of $B \cdot \text{poly}(\lambda)$ (for sufficiently large B). By using pseudo-randomness, the *client* storage in our schemes can be reduced to $\text{poly}(\lambda)$, where λ is a security parameter for computational hardness. We present our scheme generally based on a weak type of locally decodable code, which we now define.² However, we encourage readers to keep in mind the Reed-Muller based scheme mentioned in the intro. Several remarks throughout this section are designed to help in this endeavor.

Definition 5 (Locally Decodable Codes). *A locally decodable code is a tuple $(\text{Enc}, \text{Query}, \text{Dec})$ where:*

- $\text{Enc} : \Sigma^N \rightarrow \Sigma^M$ is a deterministic “encoding” procedure which maps a message m to a codeword c . N is called the message length, and M is called the block length.
- Query is a p.p.t. algorithm which on input $i \in [N]$ outputs k indices $j_1, \dots, j_k \in [M]$ along with some decoding state st .
- Dec is a p.p.t. algorithm which on input st and c_{j_1}, \dots, c_{j_k} outputs m_i .

The locally decodable code is said to be t -smooth if when sampling $(\text{st}, (j_1, \dots, j_k)) \leftarrow \text{Query}(i)$, $(j_{s_1}, \dots, j_{s_t})$ is uniformly distributed on $[M]^t$ for every distinct s_1, \dots, s_t .

The secret key is q i.i.d. uniform permutations $\pi_1, \dots, \pi_q : [M] \rightarrow [M]$, so a lower key size can be achieved at the cost of computational security by using (small-domain) pseudo-random permutations [MRS09]. A processed database $\widetilde{\text{DB}}$ is the tuple $(\text{Enc}(\text{DB}) \circ \pi_1, \dots, \text{Enc}(\text{DB}) \circ \pi_q)$, where composition of $\text{Enc}(\text{DB}) \in \Sigma^M$ with π_i denotes rearrangement of the elements of $\text{Enc}(\text{DB})$, as if $\text{Enc}(\text{DB})$ were a function mapping $[M]$ to Σ and \circ denoted function composition.

More formally, we define a scheme template as follows.

Keygen $(1^\lambda, N, B)$: Pick a t -smooth locally decodable code \mathcal{LDC} with message length N and a k -query decoding procedure, with parameters chosen so that $Bk^{t-1} \left(\frac{B}{M}\right)^{\frac{t}{2}-1} \leq 2^{-\lambda-1}$. See further discussion below on the choice of parameters.

Sample i.i.d. uniform permutations $\pi_1, \dots, \pi_k : [M] \rightarrow [M]$. Output (π_1, \dots, π_k) as the secret key. (If \mathcal{LDC} has additional parameters then they should be output as well.)

Process (sk, DB) : If necessary, encode each entry of DB as an element of Σ , so DB lies in Σ^N . Output $\widetilde{\text{DB}} = (\widetilde{\text{DB}}^{(1)}, \dots, \widetilde{\text{DB}}^{(k)})$, where $\widetilde{\text{DB}}^{(i)}$ is defined by permuting the coordinates of $\mathcal{LDC}.\text{Enc}(\text{DB})$ by π_i . That is, $\widetilde{\text{DB}}_{\pi_i(j)}^{(i)} = \mathcal{LDC}.\text{Enc}(\text{DB})_j$.

Query (sk, i) : Output $((\pi_1(j_1), \dots, \pi_k(j_k)), \text{st})$, where $((j_1, \dots, j_k), \text{st})$ is sampled according to $\mathcal{LDC}.\text{Query}(i)$.

Resp $\widetilde{\text{DB}}((\tilde{j}_1, \dots, \tilde{j}_k))$: Output $(\widetilde{\text{DB}}_{\tilde{j}_1}^{(1)}, \dots, \widetilde{\text{DB}}_{\tilde{j}_k}^{(k)})$.

Dec $(\text{st}, (\mathbf{y}_1, \dots, \mathbf{y}_k))$: Output $\mathcal{LDC}.\text{Dec}(\text{st}, (\mathbf{y}_1, \dots, \mathbf{y}_k))$

The perfect correctness of this scheme follows from the correctness of the underlying LDC.

²Our definition differs from the standard definition of locally decodable codes in that it does not require any robustness against codeword errors, and we assume that the decoding queries are non-adaptive.

The Reed-Muller Based Scheme The following polynomial-based code is a natural choice for instantiating our scheme. It is t -smooth because of the t -wise independence of degree t polynomials. Choose a finite field \mathbb{F} , integer m and a subset $H \subset \mathbb{F}$ such that $|H|^m = N$ and $|\mathbb{F}|^m = M$. For correctness, we require that $|\mathbb{F}| \geq m \cdot t \cdot (|H| - 1) + 1$.

Enc: Identify $\text{DB} \in \{0, 1\}^N$ with a map $H^m \rightarrow \{0, 1\}$ and let $\widetilde{\text{DB}} : \mathbb{F}^m \rightarrow \mathbb{F}$ be the low degree extension. Output $\widetilde{\text{DB}} \in \mathbb{F}^{\mathbb{F}^m}$.

Query: Identify $i \in [N]$ with $\mathbf{z} \in H^m$ and choose a random degree- t curve $\varphi : \mathbb{F} \rightarrow \mathbb{F}^m$ such that $\varphi(0) = \mathbf{z}$. For k at least $m \cdot t \cdot (|H| - 1)$, output the query $(\mathbf{x}_1, \dots, \mathbf{x}_k) \in \mathbb{F}^{mk}$ where $\mathbf{x}_i = \varphi(i)$.

Dec: Given responses $(a_1, \dots, a_k) \in \mathbb{F}^k$, let $\tilde{\varphi} : \mathbb{F} \rightarrow \mathbb{F}$ be the unique univariate polynomial of degree at most $k - 1$ such that $\tilde{\varphi}(i) = a_i$. Output $\tilde{\varphi}(0)$.

Let us see how the LDC constraints are satisfiable by a concrete code, and with what parameters.

Example Parameters: Low Work For (relative) simplicity, assume that $N^4 \leq B \leq \frac{2^\lambda}{\lambda^3}$. Then one can set $|H| = \lambda$, $m = \frac{\log N}{\log \lambda} (\leq \frac{\lambda}{4})$, and $|\mathbb{F}| = (2\lambda^6 B)^{1/m}$. With this choice of parameters $M = 2\lambda^6 B$, and there is a t -smooth, k -query decoding procedure (via curves) with $t = 2(\lambda + \log(2Bk) + 1)$ and $k = \lambda^3$ (as required for correctness of decoding, $|\mathbb{F}| \geq B^{1/m} \geq \lambda^4 \geq k$, and $k = \lambda^3 \geq (\lambda - 1)4(\lambda + 1)\frac{\lambda}{4} \geq (|H| - 1) \cdot m \cdot t$).

$$2Bk^{t-1} \left(\frac{B}{M}\right)^{t/2-1} = 2Bk \left(\frac{Bk^2}{M}\right)^{t/2-1} = 2Bk2^{-t/2+1} = 2^{-\lambda}$$

Thus we obtain an amortized storage overhead (defined in Section 2), server overhead, and client overheads which are all $\text{poly}(\lambda)$ (respectively λ^6 , λ^3 , and λ^3).

Example Parameters: Low Server Storage. Let $\epsilon > 0$ be any constant. Let $|H| = \max(\lambda, N^\epsilon)$ and $m = \frac{\log N}{\log |H|} (\leq \frac{1}{\epsilon})$. Suppose for simplicity that $\lambda \leq N^\epsilon \leq 2^\lambda$ and $B \leq 2^\lambda$. Let t and k be such that $t \geq 2(\lambda + \log(2Bk) + 1)$ and $k \geq m \cdot t \cdot (|H| - 1)$, which can be achieved by setting $k = O(\lambda \cdot N^\epsilon)$ and $t = O(\lambda)$. Let $|\mathbb{F}| = \max(k, (2k^2 B)^{1/m})$. With this choice of parameters $M = \max(N \cdot \lambda^{1/\epsilon}, 2k^2 B)$, which in particular is $N \cdot \text{poly}(\lambda)$ whenever $B = o\left(\frac{N^{1-2\epsilon}}{\lambda^2}\right)$. This yields

$$2Bk^{t-1} \left(\frac{B}{M}\right)^{t/2-1} = 2Bk \left(\frac{Bk^2}{M}\right)^{t/2-1} \leq 2^{-\lambda}$$

We leave further optimization of parameters (or instantiation with different locally decodable codes) to future work.

4.2.1 Proving Security

Theorem 1. *For any database size, any bound B on the number of queries, and any value λ for the security parameter, the above SSPIR scheme is a B -bounded designated client DEPIR scheme with statistical $2^{-\lambda}$ -security. Furthermore, the scheme provides database privacy.*

Proof. We show that every query (for the first B queries) is statistically close to a distribution that is simulatable given the adversary's view thus far. First, by the principle of deferred decision, we can think of the random permutations π_1, \dots, π_k as being lazily defined, input-by-input as needed. Thus the adversary's view of the ℓ^{th} query $(\pi_1(j_1), \dots, \pi_k(j_k))$ only reveals, for every $i \in [k]$, the subset of prior queries which also had $\pi_i(j_i)$ as their i^{th} coordinate. Let $S_1, \dots, S_k \subseteq [\ell - 1]$ denote these subsets.

Next, we observe that S_1, \dots, S_k inherit t -wise independence from the underlying t -smooth LDC. Furthermore, we have for each i that $S_i = \emptyset$ with probability at least $1 - \frac{B}{M}$. Our main lemma, which at this point directly implies security of our scheme, says that all such distributions are within a total variational distance ball of diameter $\epsilon = 2k^{t-1} \left(\frac{B}{M}\right)^{t/2-1}$. The advantage of an unbounded adversary is then $B\epsilon$.

Lemma 1. Let $\bar{X} = (X_1, \dots, X_k)$ and $\bar{Y} = (Y_1, \dots, Y_k)$ be t -wise independent random variables with the same marginals (i.e. for each i , X_i and Y_i are identically distributed), such that for each $i \in [k]$, there is a value \star such that $\Pr[X_i = \star] \geq 1 - \epsilon$. Then $d_{\text{TV}}(\bar{X}, \bar{Y}) \leq (k\epsilon)^{t/2} + k^{t-1}\epsilon^{t/2-1} \leq 2k^{t-1}\epsilon^{t/2-1}$

Proof. We first show that \bar{X} and \bar{Y} each have only $t/2$ non- \star values, except with probability at most $(k\epsilon)^{t/2}$.

Claim 1. $\Pr[|\{i : X_i \neq \star\}| \geq t/2] \leq (k\epsilon)^{t/2}$, and the same holds with Y_i in place of X_i .

Proof. For any $I \subset [k]$ with $|I| < t$, the probability that $X_i \neq \star$ for all $i \in I$ is at most $\epsilon^{|I|}$ by t -wise independence. The number of i for which $X_i \neq \star$ is at least $t/2$ iff for some $I \subset [k]$ with $|I| \geq t/2$, it holds that for every $i \in I$, $X_i \neq \star$. So by a union bound,

$$\Pr[|\{i : X_i \neq \star\}| \geq t/2] \leq \binom{k}{t/2} \cdot \epsilon^{t/2} \leq (k\epsilon)^{t/2}.$$

□

With this claim in hand, define G as the set of z_1, \dots, z_k for which $|\{i : z_i \neq \star\}| < t/2$. The above claim then says that $\Pr[\bar{X} \notin G]$ and $\Pr[\bar{Y} \notin G]$ are each at most $(k\epsilon)^{t/2}$. We then have

$$\begin{aligned} d_{\text{TV}}(\bar{X}, \bar{Y}) &= \frac{1}{2} \sum_{\bar{z} = z_1, \dots, z_k} \left| \Pr[\bar{X} = \bar{z}] - \Pr[\bar{Y} = \bar{z}] \right| \\ &\leq \frac{1}{2} \left(\Pr[\bar{X} \notin G] + \Pr[\bar{Y} \notin G] + \sum_{\bar{z} \in G} \left| \Pr[\bar{X} = \bar{z}] - \Pr[\bar{Y} = \bar{z}] \right| \right) \\ &\leq (k\epsilon)^{t/2} + \frac{1}{2} \sum_{\bar{z} \in G} \left| \Pr[\bar{X} = \bar{z}] - \Pr[\bar{Y} = \bar{z}] \right|. \end{aligned} \quad (1)$$

We now bound the second term of Eq. (1). We begin by rewriting, for any $\bar{z} \in G$, the event $\bar{X} = \bar{z}$ as the conjunction of two not necessarily independent events. Let I denote the set of coordinates where \bar{z} is not \star , i.e. $I \stackrel{\text{def}}{=} \{i : z_i \neq \star\}$. Then $\bar{X} = \bar{z}$ iff both

1. \bar{X} and \bar{z} agree on their restrictions to I , i.e. $\bar{X}_I = \bar{z}_I$.
2. $X_i = \star$ for every $i \notin I$. As short-hand, we write this event as $\bar{X}_{\sim I} = \star$.

We can therefore profitably rewrite $\Pr[\bar{X} = \bar{z}] = \Pr[\bar{X}_I = \bar{z}_I] \cdot \Pr[\bar{X}_{\sim I} = \star | \bar{X}_I = \bar{z}_I]$ and similarly $\Pr[\bar{Y} = \bar{z}] = \Pr[\bar{Y}_I = \bar{z}_I] \cdot \Pr[\bar{Y}_{\sim I} = \star | \bar{Y}_I = \bar{z}_I]$. By t -wise independence, the probability that $\bar{X}_I = \bar{z}_I$ is exactly the same as the probability that $\bar{Y}_i = \bar{z}_i$ (since $|I| < t/2$). Hence the difference of these probabilities has a common factor $\Pr[\bar{X}_I = \bar{z}_I] = \Pr[\bar{Y}_I = \bar{z}_I] \leq \epsilon^{|I|}$, which can be factored out to yield

$$\Pr[\bar{X}_I = \bar{z}_I] \cdot \left| \Pr[\bar{X}_{\sim I} = \star | \bar{X}_I = \bar{z}_I] - \Pr[\bar{Y}_{\sim I} = \star | \bar{Y}_I = \bar{z}_I] \right| \quad (2)$$

Claim 2. For all $I \subset [k]$ such that $|I| < t/2$, and all \bar{z}_I , it holds that

$$\begin{aligned} \left| \Pr[\bar{X}_{\sim I} = \star | \bar{X}_I = \bar{z}_I] - \Pr[\bar{Y}_{\sim I} = \star | \bar{Y}_I = \bar{z}_I] \right| &\leq 2(k - |I|)^{t-1} \epsilon^{t/2-1} \\ &\leq 2k^{t-1} \epsilon^{t/2-1}. \end{aligned}$$

Proof. All we really need about the conditional distributions $\bar{X}_{\sim I} | \bar{X}_I = \bar{z}_I$ and $\bar{Y}_{\sim I} | \bar{Y}_I = \bar{z}_I$ are that they are $t/2$ -wise independent. This $t/2$ -wise independence follows from the fact that \bar{X} and \bar{Y} are t -wise independent, and the events $\bar{X}_I = \bar{z}_I$ and $\bar{Y}_I = \bar{z}_I$ depend on fewer than $t/2$ coordinates of \bar{X} and \bar{Y} , respectively. We also only need, for each $i \notin I$, the single bit of whether or not $X_i = \star$.

So it suffices for us to prove the following slightly more abstract lemma. This lemma can be viewed as a special case of a natural generalization of Braverman's celebrated result that poly-logarithmic independence fools AC^0 [Bra09]. Instead of AC^0 our predicate is a single conjunction (with fan-in $n = k - |I|$), but the lemma does not follow directly from [Bra09] because any individual X'_i might not be uniformly distributed on $\{0, 1\}$.

Lemma 2. If (X'_1, \dots, X'_n) are t' -wise independent $\{0, 1\}$ -valued random variables such that $\mathbb{E}[X'_i] = \Pr[X'_i = 1] \geq 1 - \epsilon$ for all i , then

$$\left| \mathbb{E} \left[\prod_{i=1}^n X'_i \right] - \prod_{i=1}^n \mathbb{E}[X'_i] \right| \leq 2n(n^2\epsilon)^{t'-1}$$

Proof. For any subset $S \subseteq [n]$, write X'_S to denote the product $\prod_{i \in S} X'_i$. Let ϵ_i denote $1 - \mathbb{E}[X'_i]$. With this notation, we want to bound $\left| \mathbb{E}[X'_{[n]}] - \prod_{i=1}^n (1 - \epsilon_i) \right|$. By the binomial expansion of $X'_{[n]} = \prod_{i \in [n]} (1 - (1 - X'_i))$, we have

$$\begin{aligned} \mathbb{E}[X'_{[n]}] &= 1 + \sum_{\emptyset \neq S \subseteq [n]} \mathbb{E} \left[\prod_{i \in S} (X'_i - 1) \right] \\ &= 1 + \sum_{0 < |S| \leq t'} \mathbb{E} \left[\prod_{i \in S} (X'_i - 1) \right] + \sum_{t' < |S| \leq n} \mathbb{E} \left[\prod_{i \in S} (X'_i - 1) \right] \\ &= 1 + \sum_{0 < |S| \leq t'} \prod_{i \in S} (-\epsilon_i) + \sum_{t' < |S| \leq n} \mathbb{E} \left[\prod_{i \in S} (X'_i - 1) \right], \end{aligned} \quad (3)$$

where the last equality is by t' -wise independence. Comparing Eq. (3) to the binomial expansion

$$\prod_{i=1}^n (1 - \epsilon_i) = 1 + \sum_{0 < |S| \leq n} \prod_{i \in S} (-\epsilon_i), \quad (4)$$

we just need to bound the last term of Eq. (3) (If we generically bound the absolute value of this term, say by C , then the same bound must also apply to $\sum_{t' < |S| \leq n} \prod_{i \in S} (-\epsilon_i)$. As a result we will have $|\mathbb{E}[X'_{[n]}] - \prod_{i=1}^n (1 - \epsilon_i)| \leq 2C$). We bound

$$\begin{aligned} \left| \sum_{t' < |S| \leq n} \mathbb{E} \left[\prod_{i \in S} (X'_i - 1) \right] \right| &= \left| \mathbb{E} \left[\sum_{t' < |S| \leq n} \prod_{i \in S} (X'_i - 1) \right] \right| \\ &\leq \mathbb{E} \left[\sum_{t' < |S| \leq n} \prod_{i \in S} (X'_i - 1) \right] \\ &= \mathbb{E} \left[\sum_{\substack{S \subseteq \{i: X'_i=0\} \\ |S| > t'}} (-1)^{|S|} \right] \end{aligned} \quad (5)$$

We bound Eq. (5) via the basic fact that for any B -bounded random variable Z , we have $\mathbb{E}[Z] \leq B \cdot \Pr[Z > 0]$. We apply this fact with

$$Z = \sum_{\substack{S \subseteq \{i: X'_i=0\} \\ |S| > t'}} (-1)^{|S|}.$$

Z is 0 if $|\{i : X'_i = 0\}| \leq t'$, which happens with probability at least $1 - (n\epsilon)^{t'-1}$ by Claim 1, so $\Pr[Z > 0] \leq (n\epsilon)^{t'-1}$. When this is not the case, we bound Z via the elementary combinatorial identity $\sum_{S \subseteq [n]} (-1)^{|S|} = \sum_{i=0}^n (-1)^i \binom{n}{i} = 0$ (this follows from the binomial expansion of $(1 - 1)^n$), obtaining

$$Z = \left| \sum_{\substack{S \subseteq \{i: X'_i=0\} \\ |S| \leq t'}} (-1)^{|S|} \right| \leq \binom{n}{t'} \leq n^{t'}.$$

This implies $\mathbb{E}[Z] \leq (n\epsilon)^{t'-1} n^{t'} \leq n(n^2\epsilon)^{t'-1}$. Substituting into Eq. (5) concludes the proof of Lemma 2. \square

Claim 2 follows by applying Lemma 2 with $n = k - |I|$ and $t' = t/2$. \square

Substituting Eq. (2) into Eq. (1) and applying the following claim concludes the proof of Lemma 1. \square

Theorem 1 follows from the discussion preceding the statement of Lemma 1, and from the fact that we chose an LDC for which $2Bk^{t-1} \left(\frac{B}{M}\right)^{t/2-1} = 2^{-\lambda}$. \square

4.3 A Linear Attack if Too Many Queries are Asked

In this section we describe a technique for analyzing the query data which arises when the number of queries exceeds B . Our technique gives an attack which, in some situations, breaks the (unbounded-query) security of the (bounded-query secure) scheme from Section 4.2. Our attack utilizes extra properties of the underlying LDC which are not without loss of generality, but are nevertheless natural and satisfied by the Reed-Muller based LDC, as well as other choices based on polynomials. Roughly speaking, our attack exploits extra linear structure in the \mathcal{LDC} .Query and \mathcal{LDC} .Dec procedures. Readers familiar with polynomial-based codes will recognize the properties we use as abstractions of properties commonly used in those settings.

4.3.1 Intuition and Overview

Extra Properties of \mathcal{LDC} . Recall $\mathcal{LDC} = (\mathcal{LDC}.\text{Enc}, \mathcal{LDC}.\text{Query}, \mathcal{LDC}.\text{Dec})$. The encode and query procedures are randomized maps

$$\mathcal{LDC}.\text{Enc} : \{0, 1\}^N \rightarrow \Sigma^{[M]}; \text{ and } \mathcal{LDC}.\text{Query} : [N] \rightarrow [M]^k;$$

we assume $\mathcal{LDC}.\text{Dec}$ is a public operation and so $\mathcal{LDC}.\text{Query}$ outputs no decoding state st. We assume that $[M]$ is a vector space over some finite field \mathbb{F} , and that $[N] \subset [M]$ (not necessarily a subspace). Note in our Reed-Muller example, we had $[N] = H^m \subset \mathbb{F}^m = [M]$. Moreover, we assume there exists some subspace $V \subset [M]^k$ such that for each $i \in [N]$, $\mathcal{LDC}.\text{Query}(i)$ outputs a random element from an affine coset of V , denoted $V_i \subset [M]^k$. In the Reed-Muller case, V is the set of $(\mathbf{x}_1, \dots, \mathbf{x}_k)$ for which there exists a curve $\varphi : \mathbb{F} \rightarrow \mathbb{F}^m$ of degree at most t satisfying $\varphi(0) = 0$ and such that $\mathbf{x}_s = \varphi(s)$ for $s = 1, \dots, k$. The affine shift V_i for $i \in H^m$ is the set of \vec{x} which lie on a low degree curve satisfying $\varphi(0) = i$ instead of $\varphi(0) = 0$. We furthermore assume that for any distinct $s_1, \dots, s_{t+1} \in [k] \cup \{\text{index}\}$, there exists a linear map $\psi_{\vec{s}} : [M]^{t+1} \rightarrow [M]$ such that $\psi_{\vec{s}}(V_i|_{\vec{s}}) = i$ for all $i \in [N]$, where $V_i|_{\vec{s}}$ denotes the projection of V_i onto the coordinates \vec{s} (if $s = \text{index}$ then $V_i|_s = i$). For Reed-Muller, these linear maps are interpolation. Note this last property implies that $\mathcal{LDC}.\text{Query} : [N] \rightarrow [M]^k$ is an error-correcting code with good distance since queries to distinct $i, i' \in [N]$ can agree in at most t places. This last property also means that $\dim(V) \leq t \cdot \dim([M])$. Our scheme requires \mathcal{LDC} to be t -smooth so $\dim(V) \geq t \cdot \dim([M])$, thus $\dim(V) = t \cdot \dim([M])$. Finally, we require that \mathcal{LDC} is locally *correctable* rather than just decodeable. This means that $\mathcal{LDC}.\text{Query}$ supports queries to any codeword symbol, rather than only message symbols *i.e.*, $\mathcal{LDC}.\text{Query}$ supports any $i \in [M]$ rather than just $i \in [N]$. This means that the $|M|$ different affine planes $\{V_i\}_{i \in [M]}$, together form a subspace $\hat{V} \subset [M]^k$ of dimension $(t+1) \cdot \dim([M])$. Note that the following distributions are identical:

1. draw $i \leftarrow [M]$, $(j_1, \dots, j_k) \leftarrow V_i$, output (i, j_1, \dots, j_k) ;
2. draw $(j_1, \dots, j_k) \leftarrow \hat{V}$ and output (i, j_1, \dots, j_k) where $(j_1, \dots, j_k) \in V_i$.

This will be useful moving forward.

Our Attack. Concretely, we describe an attack which, given $B = M^{1+o(1)}$ queries $\{(j_{\alpha,1}, \dots, j_{\alpha,k})\}_{\alpha=1}^B$ from either: 1) all from V_i for a fixed $i \in [M]$; or 2) from V_{i_α} for random $i_\alpha \leftarrow [M]$, distinguishes between (1) and (2). We emphasize that this is an attack on the \mathcal{LDC} but not exactly an attack on the DEPIR scheme, because the distinguisher in the DEPIR security game only gets to see queries from V_i for $i \in [N]$ and not $i \in [M] - [N]$. This assumption is mainly to simplify the analysis; we show in Section 5.4 how to analyze the attack (in a different context) without making this assumption (but using more queries).

Notation. Since we are assuming that the query spaces, V_i , are affine planes, membership in the V_i is decided by the linear equation: $\vec{v} \in V_i$ iff $\psi_{\vec{s}}(\vec{v}) = i$ for any $\vec{s} \subset [k]$. From now on, however, we will not be explicit about the \vec{s} or the linear maps $\psi_{\vec{s}}$, we will just talk about the linear equation “ $\vec{v} \in V_i$ ”. (For the specific case of Reed-Muller codes, the coefficients of the linear equation are the Lagrange coefficients that correspond to the points where φ is evaluated.)

Intuition. The high-level idea of the attack is the following. First, initialize variables $\{\mathbf{v}_{s,j'_s}\}_{(s,j'_s) \in [k] \times [M]}$ which take values in $[M]$. The intention is that $\mathbf{v}_{s,j'_s} = j_s$ if $\pi_s(j'_s) = j_s$. For each query $(j'_{\alpha,1}, \dots, j'_{\alpha,k})$, add the constraint $\vec{v}_{j'_\alpha} \in V_i$ to a list L of linear constraints on the $\{\mathbf{v}_{s,j'_s}\}$ where $\vec{v}_{j'_\alpha}$ is shorthand for $(\mathbf{v}_{1,j'_{\alpha,1}}, \dots, \mathbf{v}_{k,j'_{\alpha,k}})$. After enough constraints have been added to L, we will be in one of two cases depending on whether the queries $(j'_{\alpha,1}, \dots, j'_{\alpha,k})$ are all to the same index or are to random indices. In the first case, there will exist non-constant assignments to the \mathbf{v}_{s,j'_s} which satisfy all constraints in L; in the second case all satisfying assignments are constant. This distinction allows the two cases to be efficiently distinguished.

The QueryDist Algorithm. Set $B = k\lambda M^{1+o(1)}$.

Input: $\{(j'_{\alpha,1}, \dots, j'_{\alpha,k})\}_\alpha$

1. Initialize variables $\{\mathbf{v}_{s,j'_s}\}_{(s,j'_s) \in [k] \times [M]}$ taking values in $[M]$, and a list L of linear constraints on the \mathbf{v}_{s,j'_s} to \emptyset . Also fix $i \in [N]$ arbitrarily.
2. For $\alpha \in \{1, \dots, B\}$, add $\vec{v}_{j'_\alpha} \in V_i$ to L, where $\vec{v}_{j'_\alpha}$ is short for $(\mathbf{v}_{1,j'_{\alpha,1}}, \dots, \mathbf{v}_{k,j'_{\alpha,k}})$.
3. Checks whether there is a non-constant assignment to the $\{\mathbf{v}_{s,j'_s}\}$ which satisfies the constraints in L. If so, output fixed, if not output random. An assignment is constant if for all (s, j'_s, j''_s) , $\mathbf{v}_{s,j'_s} = \mathbf{v}_{s,j''_s}$.

Note that QueryDist runs in time $\text{poly}(\lambda, B, M)$. Step 3 involves checking whether the space of satisfying assignments is contained in the space of constant assignments; this is possible to do efficiently using Gaussian elimination. All other steps are clearly polytime.

Lemma 3. Assume $k > 2t$, $|\mathbb{F}|^{-t} = 2^{-\Omega(\lambda)}$, and all of the assumptions mentioned in the previous paragraph. If QueryDist is given inputs $\{(j_{\alpha,1}, \dots, j_{\alpha,k})\}$ which are all queries for the same index, it outputs fixed with probability 1. If given $\{(j_{\alpha,1}, \dots, j_{\alpha,k})\}$ which are queries to random indices, it outputs random with probability $1 - 2^{-\Omega(\lambda)}$ over the queries.

Proof. If the $\{(j_{\alpha,1}, \dots, j_{\alpha,k})\}$ are all queries to i , then the assignment $\mathbf{v}_{s,j'_s} = \pi_s^{-1}(j'_s)$ is non-constant and satisfies every constraint in L since $(\pi_1^{-1}(j'_{\alpha,1}), \dots, \pi_k^{-1}(j'_{\alpha,k})) \in V_i$ for all α . If they are all queries to some other $i' \neq i$, then L can be satisfied by setting $\mathbf{v}_{s,j'_s} = (\pi'_s)^{-1}(j'_s)$ where $\pi'_s = \pi_s \circ \tau_s$ and the τ_s are any permutations such that $(\tau_1, \dots, \tau_k)(V_i) = V_{i'}$. Therefore, QueryDist outputs fixed with probability 1. We complete the proof by showing that if the $\{(j'_{\alpha,1}, \dots, j'_{\alpha,k})\}$ are queries for random indices then with overwhelming probability, any assignment to the \mathbf{v}_{s,j'_s} which satisfies all constraints in L must be constant.

Fix any assignment to the \mathbf{v}_{s,j'_s} satisfying L and for $s = 1, \dots, k$ let $\sigma_s : [M] \rightarrow [M]$ be the map which sends j'_s to the vector assigned to \mathbf{v}_{s,j'_s} . We will show that each σ_s is constant. Since the assignment satisfies L, for every $(j'_{\alpha,1}, \dots, j'_{\alpha,k})$, for all α , $(\sigma_1(j'_{\alpha,1}), \dots, \sigma_k(j'_{\alpha,k})) \in V_i$. Recall that drawing $(j'_{\alpha,1}, \dots, j'_{\alpha,k})$ consists of drawing $i_\alpha \leftarrow [N] = [M]$ and $(j_{\alpha,1}, \dots, j_{\alpha,k}) \leftarrow V_{i_\alpha}$. Therefore, every $(j_{\alpha,1}, \dots, j_{\alpha,k}) \leftarrow V_{i_\alpha}$ drawn to produce the input to QueryDist satisfies $\vec{\sigma} \circ \vec{\pi}(\vec{j}_\alpha) \in V_i$, where $\vec{\sigma} \circ \vec{\pi}(\vec{j})$ is shorthand for $(\sigma_1 \circ \pi_1(j_{\alpha,1}), \dots, \sigma_k \circ \pi_k(j_{\alpha,k}))$.

Say an assignment to $\{\mathbf{v}_{s,j'_s}\}$ is BAD if $\Pr_{(j_1, \dots, j_k) \leftarrow \hat{V}}[\vec{\sigma} \circ \vec{\pi}(\vec{j}) \in V_i] < (1 - 1/|\mathbb{F}|)$. Note,

$$\Pr_{\{\vec{j}_\alpha\}}[\exists \text{ BAD satisfying assignment}] < M^{kM} \cdot \left(1 - \frac{1}{|\mathbb{F}|}\right)^B = 2^{-\Omega(\lambda)},$$

when $B = k\lambda M^{1+o(1)}$, so we can assume that the assignment underlying σ is not BAD.

Now, fix $(j_1, \dots, j_t) \in [M]^t$ such that $\Pr_{\vec{j} \leftarrow \hat{V}}[\vec{\sigma}(\vec{j}) \in V_i | (j_1, \dots, j_t)] \geq (1 - 1/|\mathbb{F}|)$. Note drawing $\vec{j} \leftarrow \hat{V}$ conditioned on fixed (j_1, \dots, j_t) requires drawing just one more coordinate, say j_{t+1} , randomly from $[M]$ (using

the extra property as well as interpolation). Now, having fixed (j_1, \dots, j_t) , for $s \in [k]$, let $\vec{j}_s = \sigma_s \circ \pi_s(j_s)$. Let $j_{t+1}^* \in [M]$ be the unique value such that $(\vec{j}_1, \dots, \vec{j}_t, j_{t+1}^*)$ is consistent with V_i . It follows that

$$\Pr_{j_{t+1} \leftarrow [M]} [\vec{j}_{t+1} = j_{t+1}^*] \geq 1 - \frac{1}{|\mathbb{F}|}$$

and so $\sigma_{t+1} \circ \pi_{t+1}$ (hence σ_{t+1}) takes a constant value on a $(1 - 1/|\mathbb{F}|)$ -fraction of its inputs. The same is true for all of the σ_s . For each s , let $j'_s \in [M]$ be the most likely value of σ_s . We show that either $\sigma_s(j_s) = j'_s$ with probability 1 for all s , in which case the assignment is constant; or else with high probability, $\vec{\sigma} \circ \vec{\pi}(\vec{j}_\alpha) \notin V_i$ for some α , so the assignment does not satisfy L.

By the t -smoothness of \mathcal{LDC} , Claim 1 implies

$$\Pr_{\vec{j} \leftarrow \mathcal{V}} [\#\{s \in [k] : \vec{j}_s \neq j'_s\} > t] \leq (k/|\mathbb{F}|)^t = 2^{-\Omega(\lambda)}.$$

It follows that with high probability, for any α, α' ,

$$\Delta(\vec{\sigma} \circ \vec{\pi}(\vec{j}_\alpha), \vec{\sigma} \circ \vec{\pi}(\vec{j}_{\alpha'})) \leq \Delta(\vec{\sigma} \circ \vec{\pi}(\vec{j}_\alpha), \vec{j}) + \Delta(\vec{j}, \vec{\sigma} \circ \vec{\pi}(\vec{j}_{\alpha'})) \leq 2t,$$

where Δ is Hamming distance. Therefore, $\vec{\sigma} \circ \vec{\pi}(\vec{j}_\alpha)$ and $\vec{\sigma} \circ \vec{\pi}(\vec{j}_{\alpha'})$ share at least $k - 2t > t$ of their coordinates. The interpolation assumption implies $\vec{\sigma} \circ \vec{\pi}(\vec{j}_\alpha) = \vec{\sigma} \circ \vec{\pi}(\vec{j}_{\alpha'})$ for all α, α' . Therefore, σ is constant and the result follows. \square

4.4 Limits on statistical DEPIR

We show that in any B -bounded information-theoretically secure (designated client) PIR, either:

- The server's responses are almost as long as the database, or
- The length of the secret key is almost B .

The bound holds even for schemes with only imperfect correctness and security. This in particular implies that the bound holds even for schemes, like the one above, which provides information-theoretic security except for the use of pseudorandom permutations for permuting the database.

4.4.1 Answer Size in Public Client PIR

First, we show that a *public-client* PIR cannot achieve both information theoretic security and non-trivial succinctness. The intuition is that (by statistical security) a single query is indistinguishable from a query to any other index, so the server's answer contains almost all the information about the database. Let $|a|$ denote the size of a server's response – i.e. the length of an output of `Resp`.

Theorem 2. *In any statistically δ -secure public-client DEPIR with correctness error ϵ :*

$$|a| \geq (1 - H(\epsilon + \delta))N - O(\log N).$$

In fact, this lower bound holds even if the PIR is only non-adaptively δ -secure.

Proof. Suppose the PIR is non-adaptively δ -secure. Then

Claim 3. *There is an algorithm `Reconstruct` such that for any database DB ,*

$$\mathbb{E} [\Delta(DB', DB)] \geq (1 - \epsilon - \delta)N$$

in the probability space defined by sampling

$$\begin{aligned} (\text{pk}, \text{sk}) &\leftarrow \text{Keygen}(1^\lambda, N) \\ DB &\leftarrow \text{Process}(\text{sk}, DB) \\ (q, \text{st}) &\leftarrow \text{Query}(\text{pk}, 0) \\ a &\leftarrow \text{Resp}(q, DB) \\ DB' &\leftarrow \text{Reconstruct}(\text{pk}, q, a). \end{aligned}$$

Proof. Reconstruct is an algorithm which does the following: Given (pk, q, a) , it samples st_1, \dots, st_N , where each st_i is independently sampled from the distribution of the state st obtained by sampling $(q', st) \leftarrow \text{Query}(pk, i)$ conditioned on $q' = q$. Finally Reconstruct outputs DB' such that for each i , $DB'_i = \text{Dec}(st_i, a)$. By linearity of expectation, it suffices to show that for a uniformly random i , $\Pr[DB_i = \text{Dec}(st_i, a)] \geq 1 - \epsilon - \delta$.

By the non-adaptive δ -security, if we instead sample $(q, st) \leftarrow \text{Query}(pk, i)$, this change modifies the distribution of (\widetilde{DB}, pk, q) by statistical distance at most δ . Since a is a function of (pk, q) , and st_i is a function of (pk, q, a) , the distribution of $(\widetilde{DB}, a, st_i)$ and therefore the probability that $DB_i = \text{Dec}(st_i, a)$ also changes by at most δ . However, the modified experiment induces a distribution of $(\widetilde{DB}, pk, q, st_i)$ which is exactly as if one had sampled $(q, st_i) \leftarrow \text{Query}(pk, i)$. Thus in the modified experiment, $\Pr[DB_i = \text{Dec}(st_i, a)] \geq 1 - \epsilon$, which means that in the original experiment $\Pr[DB_i = \text{Dec}(st_i, a)] \geq 1 - \epsilon - \delta$. \square

Claim 4. A random variable $X \in \{0, 1\}^N$ with expected Hamming weight is at most ϵN (with $\epsilon \leq \frac{1}{2}$) has entropy at most $H(\epsilon)N + O(\log N)$.

Proof. We can partition the set of N -bit strings based on their Hamming weight. It is clear that the maximal entropy of X is achieved only when the distribution of X is uniform within each class. In other words, an entropy-maximizing distribution takes the form $\sum_{i=0}^N w_i U_i$, where U_i is the uniform distribution on strings with Hamming weight i and $\{w_i\}$ are non-negative weights summing to 1. The entropy of this distribution is $-\sum_i w_i \log w_i + \sum_i w_i H(U_i)$, which is at most $\sum_i w_i H(U_i) + \log(N+1)$. So we want to maximize $\sum_i w_i H(U_i) = \sum_i w_i \log \binom{N}{i}$ subject to the constraint that $\sum_i w_i \cdot i \leq \epsilon N$.

We have the bound

$$\begin{aligned} \sum_i w_i \log \binom{N}{i} &= O(\log N) + \sum_i w_i H\left(\frac{i}{N}\right) N && \text{by Stirling's formula} \\ &\leq O(\log N) + H\left(\sum_i \frac{w_i \cdot i}{N}\right) N && \text{by Jensen's inequality} \\ &\leq O(\log N) + H(\epsilon)N && \text{because } \sum_i \frac{w_i \cdot i}{N} \leq \epsilon \leq \frac{1}{2}. \quad \square \end{aligned}$$

To prove the theorem, we use the following basic fact. If X and Y are random variables, then

$$0 \leq H(X) - H(X|Y) \leq H(Y).$$

We apply this where $X = DB$ is uniformly random (i.e. has entropy N), and $Y = a$. Conditioning on the value of a can reduce the entropy of DB by at most $|a|$. The fact that Reconstruct produces DB' which agrees in expectation with DB on at least $(1 - \epsilon - \delta)N$ locations implies that $H(DB|a) \leq H(\epsilon + \delta)N + O(\log N)$. Thus

$$|a| \geq N - (H(\epsilon + \delta)N + O(\log N)) = (1 - H(\epsilon + \delta))N - O(\log N),$$

which concludes the proof of Theorem 2 \square

4.4.2 Key Size for Designated Client PIR

The idea behind our lower bound on key size for a designated client scheme is that releasing the key turns the scheme into a public client scheme, which by Theorem 2 must be insecure. Since the key is a piece of information which causes a dramatic change in the entropy of a random variable, it must be long.

Theorem 3. In any B -bounded-query statistically δ_{dc} -secure, $(1 - \epsilon)$ -correct designated-client DEPIR, the key size $|sk_{dc}|$ satisfies:

$$|sk_{dc}| \geq \left(1 - 2\delta_{dc} - H\left(\frac{1 - \delta_{pc}}{2}\right)\right) \cdot B - O(\log B)$$

where δ_{pc} is the smallest δ such that $H(\epsilon + \delta) \geq 1 - \frac{|a| + O(\log N)}{N}$. In particular, if $|a| = o(N)$, $\epsilon = o(1)$, and $\delta_{dc} = o(1)$, then $|sk_{dc}| = \Omega(B)$.

Proof. Given any designated client PIR scheme with correctness error ϵ , answer size $|a|$, and B -query δ_{dc} -security, we can turn it into a public-client scheme with the same correctness probability and answer size $|a|$ by just releasing the secret key. By Theorem 2, this scheme can only be non-adaptively δ -secure if

$$H(\epsilon + \delta) \geq 1 - \frac{|a| + O(\log N)}{N}$$

Let δ_{pc} denote the minimum such δ . Note that if $\epsilon = o(1)$ and $|a| = o(N)$, then $\delta_{\text{pc}} = \frac{1}{2} - o(1)$. By definition of δ_{pc} , there exists a database DB and indices i_0 and i_1 such that the distributions $(\text{pk}, \widetilde{\text{DB}}, q_0)$ and $(\text{pk}, \widetilde{\text{DB}}, q_1)$ are δ_{pc} -far (i.e. distinguishable with probability $\frac{1+\delta_{\text{pc}}}{2}$) in the probability space defined by sampling

$$\begin{aligned} (\text{pk}_{\text{pc}}, \text{sk}_{\text{pc}}) &\leftarrow \text{Keygen}(1^\lambda, N) \\ \widetilde{\text{DB}} &\leftarrow \text{Process}(\text{sk}_{\text{pc}}, \text{DB}) \\ q_0 &\leftarrow \text{Query}(\text{pk}, i_0) \\ q_1 &\leftarrow \text{Query}(\text{pk}, i_1). \end{aligned}$$

In particular, suppose a uniformly random string $M \in \{0, 1\}^B$ is chosen, and an adversary \mathcal{A} is given $(\text{pk}_{\text{pc}}, \widetilde{\text{DB}}, q_1, \dots, q_B)$, where $q_j \leftarrow \text{Query}(\text{pk}, i_{m_j})$. Then it is possible for \mathcal{A} to output M' such that in expectation, $\Delta(m, m') \geq \frac{1+\delta_{\text{pc}}}{2} \cdot B$. In other words,

$$H(M | (\text{pk}_{\text{pc}}, \widetilde{\text{DB}}, q_1, \dots, q_B)) \leq H\left(\frac{1-\delta_{\text{pc}}}{2}\right) B + O(\log B).$$

On the other hand, we can use the following lemma, proved implicitly by Bellare et al. [BTV12] and explicitly restated by Dodis [Dod12] to lower bound the entropy of M given only $\widetilde{\text{DB}}, q_1, \dots, q_B$.

Lemma 4 ([BTV12]). *For any (possibly correlated) distributions M, C over some spaces \mathcal{M} and \mathcal{C} , let*

$$\epsilon = \mathbf{SD}((M, C); M \times C)$$

where $M \times C$ is the product distribution of the independent marginal distributions M and C . Then,

$$2\epsilon^2 \leq \mathbf{I}(M; C) \leq 2\epsilon \cdot \log(|\mathcal{M}|/\epsilon)$$

This lemma, together with the B -bounded δ_{dc} -security of our designated client PIR, implies that

$$\begin{aligned} H(M | \widetilde{\text{DB}}, q_1, \dots, q_B) &\geq B - 2\delta_{\text{dc}}(B - \log \delta_{\text{dc}}) \\ &= (1 - 2\delta_{\text{dc}})B + 2\delta_{\text{dc}} \log \delta_{\text{dc}} \\ &\geq (1 - 2\delta_{\text{dc}})B - 2. \end{aligned}$$

$|\text{pk}_{\text{pc}}| = |\text{sk}_{\text{dc}}|$ must be large as the difference between these two entropies, which proves the theorem. \square

5 A Candidate Designated Client Scheme and HPN

In this section we propose a candidate designated-client DEPIR scheme for unbounded number of queries. We prove the security of the scheme based on the hardness of a new computational problem, called the hidden permutation with noise problem (HPN).

Our candidate scheme is similar to the bounded secure scheme of Section 4, except that the client adds noise to each of its queries by overwriting some of the coordinates with random values. This modification thwarts the linear attack of last section; however, due to the impossibility of statistically secure designated client DEPIR (see Section 4.4), the security of a scheme that follows these lines can only be computational *even if the client uses perfectly random permutations*. Thus, an additional hardness assumption is necessary.

We formulate a number of variants of HPN. The strongest variant is essentially a restatement of the privacy requirement from the scheme, as per Definition 1. It says that no PPT \mathcal{A} can distinguish the queries generated by the

scheme from sequences of random values in \mathbb{F}^m , even if \mathcal{A} has full adaptive control over which addresses are queried. We then give reductions from the adaptive to static versions of HPN and from the decision version to a search version where hidden random permutations are recovered in full. This implies that our scheme is secure as long as the static, search problem remains hard.

The candidate scheme is presented in Section 5.1, the HPN assumption is defined in Section 5.2, and the reductions are proved in Sections 5.3 and 5.4.

5.1 Candidate Scheme

Notation. Let \mathcal{P} be a family of pseudorandom permutations with seed length λ . We use the same algebraic notation as in the Introduction. So let $(m, t, r, k, H, \mathbb{F})$ be such that $m < t < r < k < |\mathbb{F}|$ and $H \subset \mathbb{F}$ such that $|H|^m = N$ and $m(t-1)(|H|-1) < r$. Also, $|\mathbb{F}|^m = \text{poly}(n)$ while $|\mathbb{F}|^t = n^{\omega(1)}$.

Keygen $(1^\lambda, N)$: Draw permutations $\tau \leftarrow \mathcal{P}(\mathbb{F}^{m+1})$ and $\pi \leftarrow \mathcal{P}(\mathbb{F}^m)$ and a subset $T \subset [k]$ of size $|T| = r$. Output $k = (\text{seed}_\pi, \text{seed}_\tau, T)$.

Process (k, DB) : Interpret $\text{DB} \in \{0, 1\}^N$ as $\text{DB} : H^m \rightarrow \{0, 1\}$ using the identification of $[N]$ with (a subset of) H^m (DB is zero on any points in H^m not in the image of this identification) and let $\widehat{\text{DB}} : \mathbb{F}^m \rightarrow \mathbb{F}$ be the low degree extension, so $\deg(\widehat{\text{DB}}) \leq m(|H|-1)$. Output

$$\widetilde{\text{DB}} = \left\{ \left(\mathbf{x}, \tau(\mathbf{x}, \widehat{\text{DB}}(\pi^{-1}(\mathbf{x}))) \right) \right\}_{\mathbf{x} \in \mathbb{F}^m}.$$

Query (k, i) : Let $\mathbf{z} \in H^m$ be the element corresponding to $i \in [N]$. Choose $\vec{\mathbf{x}} = (\mathbf{x}_1, \dots, \mathbf{x}_k) \leftarrow \mathbf{V}_{\mathbf{z}}$. Output $\vec{\mathbf{y}} = (\mathbf{y}_1, \dots, \mathbf{y}_k)$ where $\mathbf{y}_i = \pi(\mathbf{x}_i)$ if $i \in T$, and $\mathbf{y}_i \leftarrow \mathbb{F}^m$ is random otherwise. Set $\text{st} = (\text{seed}_\tau, T)$.

Resp $\widetilde{\text{DB}}(q)$: Upon receiving $q = (\mathbf{y}_1, \dots, \mathbf{y}_k)$, for each $i = 1, \dots, k$, find the row of $\widetilde{\text{DB}}$ with first coordinate \mathbf{y}_i : $(\mathbf{y}_i, \mathbf{y}'_i)$. Output $a = (\mathbf{y}'_1, \dots, \mathbf{y}'_k)$.

Dec (st, a) : Parse $a = (\mathbf{y}'_1, \dots, \mathbf{y}'_k)$ and $\text{st} = (\text{seed}_\tau, T)$. For $i \in T$, set $(\mathbf{x}_i, \alpha_i) = \tau_i^{-1}(\mathbf{y}'_i)$. Let $\psi : \mathbb{F} \rightarrow \mathbb{F}$ be the unique univariate polynomial of degree at most $m(t-1)(|H|-1)$ such that $\psi(i) = \alpha_i$ for all $i \in T$. Output \perp if no such polynomial exists; otherwise output $\psi(0)$.

5.2 Variants of HPN

All variants of the HPN problem are defined in terms of the the HPN distribution, which is essentially a noisy version of the distributions $\mathcal{D}(\{\pi_i\}, \mathbf{z})$ from the previous section. Recall this distribution: $\vec{\mathbf{x}} \leftarrow \mathbf{V}_{\mathbf{z}}$ is drawn and $\vec{\mathbf{y}}$ is output where $\mathbf{y}_i = \pi_i(\mathbf{x}_i)$. In this section we simplify notations and use the same permutation π for all coordinates $i \in [k]$. We add noise to the samples in the following way: at the beginning of the experiment a random subset $T \subset [k]$ of size $r > t$ is chosen and $\mathbf{y}_i = \pi(\mathbf{x}_i)$ only if $i \in T$; otherwise \mathbf{y}_i is drawn randomly from \mathbb{F}^m . Intuitively, HPN says that no PPT adversary can either distinguish such samples from random (decision form) or recover the hidden permutation π (search version).

Definition 6 (HPN Distribution). Let (m, t, r, k, \mathbb{F}) be such that $m < t < r < k < |\mathbb{F}|$, $|\mathbb{F}|^m = \text{poly}(n)$ and $|\mathbb{F}|^t = n^{\omega(1)}$. For $\pi \in \text{Perm}(\mathbb{F}^m)$, $\mathbf{z} \in \mathbb{F}^m$ and $T \subset [k]$, let $\mathcal{D}(\pi, \mathbf{z}, T)$ be the distribution which: draws $\vec{\mathbf{x}} \leftarrow \mathbf{V}_{\mathbf{z}}$ and outputs $\vec{\mathbf{y}} \in \mathbb{F}^{mk}$ where $\mathbf{y}_i = \pi(\mathbf{x}_i)$ if $i \in T$ and $\mathbf{y}_i \leftarrow \mathbb{F}^m$ otherwise.

The static versions of the HPN assumption are formulated against a PPT adversary who gets polynomially many samples $(\mathbf{z}_\alpha, \vec{\mathbf{y}}_\alpha)$ where $\vec{\mathbf{y}}_\alpha \leftarrow \mathcal{D}(\pi, \mathbf{z}_\alpha, T)$ for random fixed $\pi \in \text{Perm}(\mathbb{F}^m)$ and $T \subset [k]$ and random $\mathbf{z}_\alpha \leftarrow \mathbb{F}^m$. In the adaptive versions of the assumption allow the adversary to choose the $\mathbf{z}_\alpha \in \mathbb{F}^m$ adaptively as the experiment

progresses. The static versions of HPN imply the adaptive versions as $|\mathbb{F}|^m = \text{poly}(n)$, and so a static adversary who receives enough random samples, will be able to provide samples to an adaptive adversary. This idea is utilized in the proof of Claim 5 below. In the definitions below we write $\{\vec{y}_\alpha\} \leftarrow \mathcal{D}(\pi, \{\mathbf{z}_\alpha\}, T)$ for the samples; it is to be understood that the indices $\{\mathbf{z}_\alpha\}$ are chosen randomly from \mathbb{F}^m and made public to the adversary.

Definition 7 (The HPN Assumption). *Let (m, t, r, k, \mathbb{F}) be as above.*

Search Version: *For all PPT algorithms \mathcal{A} and non-negligible $\delta > 0$,*

$$\Pr_{\pi, T, \{\vec{y}_\alpha\}} \left[\mathcal{A}(\{\vec{y}_\alpha\}) = \pi \right] < \delta.$$

The probability above is over $\pi \leftarrow \text{Perm}(\mathbb{F}^m)$, random subset $T \subset [k]$ such that $|T| = r$, and $\{\mathbf{z}_\alpha\} \leftarrow \mathbb{F}^m$, $\{\vec{y}_\alpha\} \leftarrow \mathcal{D}(\pi, \{\mathbf{z}_\alpha\}, T)$.

Decision Version: *For all PPT \mathcal{A} and non-negligible $\delta > 0$,*

$$\Pr_{\pi, T} \left[\left| \Pr_{\{\vec{y}_\alpha\} \leftarrow \mathcal{D}(\pi, \{\mathbf{z}_\alpha\}, T)} [\mathcal{A}(\{\vec{y}_\alpha\}) = 1] - \Pr_{\{\vec{y}_\alpha\} \leftarrow \mathbb{F}^{km}} [\mathcal{A}(\{\vec{y}_\alpha\}) = 1] \right| > \delta \right] < \delta,$$

where the outer probability is over $\pi \leftarrow \text{Perm}(\mathbb{F}^m)$ and random $T \subset [k]$ such that $|T| = r$.

Claim 5. *The candidate scheme is secure assuming decisional HPN.*

Proof. The ideal world simulator for the candidate scheme simply sends a random $\vec{y}_\alpha \in \mathbb{F}^{km}$ anytime the adversary \mathcal{A} requests a query for some index \mathbf{z}_α . Note that decisional HPN says that the simulated transcript is indistinguishable from valid queries to a random sequence of address vectors $\mathbf{z}_\alpha \in \mathbb{F}^m$. This doesn't quite prove security since the DEPIR chooses arbitrary addresses in H^m in an adaptive fashion. However, note that since $|\mathbb{F}|^m = \text{poly}(n)$, arbitrary adaptive address choice does not grant extra power to the adversary. Indeed, if \mathcal{A} requests a query to a particular address $\mathbf{z} \in H^m$, a distinguisher who gets queries corresponding to random $\mathbf{z}_\alpha \in \mathbb{F}^m$ simply asks for $|\mathbb{F}|^m \cdot n$ such queries and forwards to \mathcal{A} the first one for which $\mathbf{z}_\alpha = \mathbf{z}$. With high probability, some such α will exist. \square \square

5.3 Search to Decision Reduction

Theorem 4. *If there exists a PPT \mathcal{A} which breaks the decisional HPN–assumption, then there exists PPT \mathcal{B} which breaks search HPN.*

Proof. Let \mathcal{A} be any PPT algorithm and $\delta > 0$ non-negligible such that

$$\Pr_{\pi, T} \left[\left| \Pr_{\{\vec{y}_\alpha\} \leftarrow \mathcal{D}(\pi, \{\mathbf{z}_\alpha\}, T)} [\mathcal{A}(\{\vec{y}_\alpha\}) = 1] - \Pr_{\{\vec{y}_\alpha\} \leftarrow \mathbb{F}^{km}} [\mathcal{A}(\{\vec{y}_\alpha\}) = 1] \right| > \delta \right] \geq \delta.$$

We construct \mathcal{B} which recovers π given samples from $\mathcal{D}(\pi, \{\mathbf{z}_\alpha\}, T)$ and oracle access to \mathcal{A} . Our algorithm \mathcal{B} will proceed in two steps. First, it will use \mathcal{A} to recover a large (size at least $t + 1$) subset $T' \subset T$. We call this the “cleaning step” since \mathcal{B} removes all noise from the queries: given $\vec{y} \leftarrow \mathcal{D}(\pi, \mathbf{z}, T)$, $(\vec{y})_{T'}$ consists entirely of correct permuted evaluations of a curve. Then, \mathcal{B} passes the cleaned samples to the RecoverPerm algorithm, described and analyzed in Section 5.4. RecoverPerm outputs π given polynomially many noiseless samples, and is an extension of the distinguishing attack on the bounded scheme from Section 4.3. We now describe the CLEAN algorithm. \square

The Distribution \mathcal{D}_S . For $S \subset [k]$, let \mathcal{D}_S be the distribution which draws $\{\vec{y}_\alpha\} \leftarrow \mathcal{D}(\pi, \{\mathbf{z}_\alpha\}, T)$, and outputs $\{\vec{y}'_\alpha\}$ where

$$\mathbf{y}'_{\alpha, i} = \begin{cases} \mathbf{y}_{\alpha, i}, & i \in S \\ \mathbf{y}'' \leftarrow \mathbb{F}^m, & i \notin S \end{cases}$$

Remark. We make the following observations about \mathcal{D}_S .

1. Since \mathcal{B} is given samples from $\mathcal{D}(\pi, \{\mathbf{z}_\alpha\}, T)$, it can efficiently obtain samples from \mathcal{D}_S for any $S \subset [k]$. Since \mathcal{B} additionally gets oracle access to \mathcal{A} , it can approximate

$$p_S := \Pr_{\{\vec{\mathbf{y}}_\alpha\} \leftarrow \mathcal{D}_S} [\mathcal{A}(\{\vec{\mathbf{y}}_\alpha\}) = 1]$$

to within arbitrary inverse polynomial accuracy in polynomial time with probability $1 - 2^{-n}$.

2. For all $S \subset [k]$, \mathcal{D}_S is identically distributed to $\mathcal{D}_{S \cap T}$. In particular, if $i \notin T$ then $\mathcal{D}_S \equiv \mathcal{D}_{S \cup \{i\}}$.
3. If $S = [k]$ then $\mathcal{D}_S = \mathcal{D}(\pi, \{\mathbf{z}_\alpha\}, T)$; if $|S| \leq t$ then \mathcal{D}_S is the uniform distribution on $\mathbb{F}^{mk \cdot \text{poly}}$ by t -wise independence of degree t curves.

Intuition. The main idea of CLEAN is the following. It samples $\{\vec{\mathbf{y}}_\alpha\} \leftarrow \mathcal{D}(\pi, \{\mathbf{z}_\alpha\}, T)$, chooses $i \in [k]$ and replaces every $\mathbf{y}_{\alpha,i}$ with a random element of \mathbb{F}^m . If $i \notin T$, then the distribution of $\{\vec{\mathbf{y}}_\alpha\}$ has not changed, if $i \in T$, then each $\vec{\mathbf{y}}_\alpha$ has one fewer correct coordinate. If CLEAN was lucky in the choice of i , \mathcal{A} will change its decision probability, and CLEAN decides that $i \in T$. A hybrid argument shows that if CLEAN will efficiently be able to find enough “lucky” $i \in T$, to output $T' \subset T$ of size at least $t + 1$.

The CLEAN Algorithm. For a parameter $\delta > 0$, set $M = nk^2/\delta^2$ and let c be a universal constant.

1. Use M samples from $\mathcal{D}_{[k]}$ and \mathcal{D}_\emptyset to compute $\hat{p}_{[k]}$ and \hat{p}_\emptyset , approximations of $p_{[k]}$ and p_\emptyset .
2. **Initialize** $T' = \emptyset$; **While** $|T'| \leq t$:
 - **Initialize** $S = T'$, $\hat{p}_S = \hat{p}_\emptyset$; **While** $S \neq [k]$:
 - * Pick $i \notin S$. Use M samples from $\mathcal{D}_{S \cup \{i\}}$ to compute $\hat{p}_{S \cup \{i\}}$, an approximation of $p_{S \cup \{i\}}$.
 - * If $|\hat{p}_{S \cup \{i\}} - \hat{p}_S| > \delta/2k$, $T' = T' \cup \{i\}$.
 - * Redefine $S = S \cup \{i\}$, $\hat{p}_S = \hat{p}_{S \cup \{i\}}$.
3. **Output** T' .
4. **Time Out Condition:** If the total runtime ever reaches 2^{cn} , abort and output T' .

Lemma 5. Let $(m, t, r, k, |\mathbb{F}|)$ be as above. Suppose PPT \mathcal{A} , non-negligible $\delta > 0$ and $\pi \in \text{Perm}(\mathbb{F}^m)$, $T \subset [k]$ of size $|T| = r$ are such that

$$\left| \Pr_{\{\vec{\mathbf{y}}_\alpha\} \leftarrow \mathcal{D}(\pi, \{\mathbf{z}_\alpha\}, T)} [\mathcal{A}(\{\vec{\mathbf{y}}_\alpha\}) = 1] - \Pr_{\{\vec{\mathbf{y}}_\alpha\} \leftarrow \mathbb{F}^{km}} [\mathcal{A}(\{\vec{\mathbf{y}}_\alpha\}) = 1] \right| > \delta.$$

Then

$$\Pr_{\{\vec{\mathbf{y}}_\alpha\} \leftarrow \mathcal{D}(\pi, \{\mathbf{z}_\alpha\}, T)} \left[\text{CLEAN}^{\mathcal{A}}(\{\vec{\mathbf{y}}_\alpha\}) = T' \text{ st } T' \subset T \text{ and } |T'| = t + 1 \right] \geq 1 - 2^{-\Omega(n)}.$$

Moreover, the expected running time of CLEAN is $\mathcal{O}(nk^3t/\delta^2)$.

Proof. We show that if all approximations \hat{p} computed by CLEAN are within $\delta/4k$ of the true expectations p (in this case we say that the approximations are *good*) then CLEAN runs in $\mathcal{O}(Mkt)$ time and outputs $T' \subset T$ of size at least $t + 1$. This completes the proof since by the Chernoff-Hoeffding inequality, all of CLEAN’s approximations are good with probability $1 - 2^{-\Omega(n)}$; the maximum runtime is 2^{cn} because of the time-out condition, so events which occur with probability $2^{-\Omega(n)}$ may be safely ignored. Note that when CLEAN’s approximations are good, the output T' is a subset of T . Indeed, the only way i gets added to T' is if $|\hat{p}_{S \cup \{i\}} - \hat{p}_S| > \delta/2k$; if $i \notin T$ and approximations are good:

$$|\hat{p}_{S \cup \{i\}} - \hat{p}_S| \leq |\hat{p}_{S \cup \{i\}} - p_{S \cup \{i\}}| + |p_{S \cup \{i\}} - p_S| + |\hat{p}_S - p_S| \leq \frac{\delta}{2k} + |p_{S \cup \{i\}} - p_S| = \frac{\delta}{2k},$$

by Observation 2. We show that if approximations are good then each time through the outer while loop at least one i is added to T' . This completes the proof since the time to run the inner while loop is $\mathcal{O}(Mk)$. The key point is that throughout the course of the inner loop S goes from $|S| \leq t$ (so that $p_S = p_\emptyset$ by Observation 3) to $S = [k]$. Since \mathcal{A} distinguishes $\mathcal{D}(\{\pi_i\}, \{\mathbf{z}_\alpha\}, T)$ from uniform with probability at least $\delta > 0$, $|p_{[k]} - p_\emptyset| \geq \delta$. By a hybrid argument, there must exist some (S, i) encountered during the course of the inner loop so that $|p_{S \cup \{i\}} - p_S| \geq \delta/k$. Since approximations are good, $|\hat{p}_{S \cup \{i\}} - \hat{p}_S| \geq \delta/2k$ and so i is added to T' . \square \square

5.4 Recovering the Permutations

In this section we describe a protocol which is given samples from $\mathcal{D}(\pi, \{\mathbf{z}_\alpha\})$ and recovers the permutation π used to produce the samples. This attack extends to the case of many permutations $\{\pi_i\}$ instead of just one, and so constitutes a strong break of the unbounded variant of the scheme from Section 4. We use the same notation as in that section. This algorithm requires more samples than the one in the previous section ($\text{poly}(|\mathbb{F}|^m)$ versus $|\mathbb{F}|^{m(1+o(1))}$).

The RecoverPerm Algorithm. Let $(m, t, k, |\mathbb{F}|)$ be such that $m < t < k < |\mathbb{F}|$ and $|\mathbb{F}|^m = \text{poly}(\lambda)$. Set $B = m\lambda|\mathbb{F}|^{3m+3}$. Let $\{\mathbf{z}_\alpha\}$ be any sequence of addresses such that every $\mathbf{z} \in H^m$ appears at least B times in $\{\mathbf{z}_\alpha\}$.

Input: $\{\vec{\mathbf{y}}_\alpha\}$ drawn from $\mathcal{D}(\pi, \{\mathbf{z}_\alpha\})$.

1. Initialize variables $\{\mathbf{v}_\mathbf{y}\}_{\mathbf{y} \in \times \mathbb{F}^m}$ taking values in \mathbb{F}^m , and a list L of linear constraints to \emptyset .
2. For all α , add the constraint $\vec{\mathbf{v}}_{\vec{\mathbf{y}}_\alpha} \in V_{\mathbf{z}_\alpha}$ to L , where $\vec{\mathbf{v}}_{\vec{\mathbf{y}}_\alpha}$ is shorthand for the vector $(\mathbf{v}_{\mathbf{y}_{\alpha,1}}, \dots, \mathbf{v}_{\mathbf{y}_{\alpha,k}})$.
3. If the constraints in L are inconsistent (*i.e.* no assignment to the $\mathbf{v}_\mathbf{y}$ satisfies all constraints), abort and output \perp . Otherwise, choose an arbitrary assignment satisfying all constraints in L and let $\sigma : \mathbb{F}^m \rightarrow \mathbb{F}^m$ be the map which sends \mathbf{y} to the vector assigned to $\mathbf{v}_\mathbf{y}$. If σ is not a permutation, abort and output \perp . Otherwise, output $\pi' \in \text{Perm}(\mathbb{F}^m)$ where $\pi' = \sigma^{-1}$.

Note that RecoverPerm runs in time $\text{poly}(B, |\mathbb{F}|^m, \lambda)$ since Step 3 involves solving a system of linear equations and checking whether a functions with polysized domain is a permutation; all other steps are efficient. The next Lemma states RecoverPerm recovers the correct π used to generate the samples with high probability.

Lemma 6. Set $B = m\lambda|\mathbb{F}|^{3m+3}$ and let $\{\mathbf{z}_\alpha\}$ be any sequence of addresses such that each $\mathbf{z} \in H^m$ appears at least B times in $\{\mathbf{z}_\alpha\}$. Let $\pi \in \text{Perm}(\mathbb{F}^m)$ be the permutation used to generate the input samples, and let $\pi' \in \text{Perm}(\mathbb{F}^m)$ the output. Then with overwhelming probability: $\pi' = \pi$.

Proof. The first failure event in Step 3, that the linear constraints in L are inconsistent, never occurs as by definition $\mathbf{v}_\mathbf{y} = \pi^{-1}(\mathbf{y})$ satisfies every constraint. Fix any assignment to the $\mathbf{v}_\mathbf{y}$ satisfying the constraints in L , let $\sigma : \mathbb{F}^m \rightarrow \mathbb{F}^m$ be the map which sends \mathbf{y} to the vector assigned to $\mathbf{v}_\mathbf{y}$. Since the assignment satisfies L , for every $\vec{\mathbf{y}}_\alpha$, we have $\sigma(\vec{\mathbf{y}}_\alpha) \in V_{\mathbf{z}_\alpha}$, where $\sigma(\vec{\mathbf{y}}_\alpha)$ is shorthand for $(\sigma(\mathbf{y}_{\alpha,1}), \dots, \sigma(\mathbf{y}_{\alpha,k}))$. Recall that drawing $\vec{\mathbf{y}}_\alpha \leftarrow \mathcal{D}(\pi, \mathbf{z}_\alpha)$ consists of drawing $\vec{\mathbf{x}}_\alpha \leftarrow V_{\mathbf{z}_\alpha}$ and then setting $\vec{\mathbf{y}}_\alpha = \pi(\vec{\mathbf{x}}_\alpha)$. Therefore, for every $\vec{\mathbf{x}}_\alpha \leftarrow V_{\mathbf{z}_\alpha}$ drawn to produce the input of RecoverPerm: $\sigma \circ \pi(\vec{\mathbf{x}}_\alpha) \in V_{\mathbf{z}_\alpha}$.

We say that an assignment to the $\mathbf{v}_\mathbf{y}$ is BAD if $\Pr_{\vec{\mathbf{x}} \leftarrow V_{\mathbf{z}}}[\sigma \circ \pi(\vec{\mathbf{x}}) \in V_{\mathbf{z}}] < 1 - |\mathbb{F}|^{-(2m+2)}$, for some $\mathbf{z} \in H^m$. Note,

$$\Pr[\exists \text{ BAD satisfying asst}] < |\mathbb{F}|^m |\mathbb{F}|^m \cdot \left(1 - \frac{1}{|\mathbb{F}|^{2m+2}}\right)^B = 2^{-\Omega(\lambda)},$$

when $B = m\lambda|\mathbb{F}|^{3m+3}$. Therefore, it suffices to assume that the assignment is not BAD. Lemma 7 below shows that in this case, $\sigma \circ \pi = \mathbb{1}$. The result follows. \square \square

Lemma 7. Suppose $f_1, \dots, f_k : \mathbb{F}^m \rightarrow \mathbb{F}^m$ are such that $\Pr_{\vec{\mathbf{x}} \leftarrow V_{\mathbf{z}}}[\vec{f}(\vec{\mathbf{x}}) \in V_{\mathbf{z}}] \geq 1 - |\mathbb{F}|^{-(2m+2)}$ for all $\mathbf{z} \in H^m$. Then there exists a curve $\varphi : \mathbb{F} \rightarrow \mathbb{F}^m$ of degree at most t satisfying $\varphi(0) = \mathbf{0}$ such that $f_i(\mathbf{x}_i) = \mathbf{x}_i + \varphi(i)$ for all $i = 1, \dots, k$. In particular, if all f_i are equal then each f_i is the identity function.

Proof. The second statement follows from the first since if $\varphi(i)$ takes the same value for $i = 1, \dots, k$ then φ must be constant, hence identically zero.

The proof of the first statement consists of three steps, described momentarily. First however we define the following random variables. Let \vec{f} be functions that satisfy the lemma hypotheses. For any $\vec{c} \in V_0$, define $\vec{f}_{\vec{c}}(\vec{x}) := \vec{f}(\vec{x} + \vec{c}) - \vec{f}(\vec{c})$. Choose $\vec{c} \in V_0$ so that $\vec{f}_{\vec{c}}(\mathbf{0}) = \mathbf{0}$ and $\Pr_{\vec{x} \leftarrow V_z} [\vec{f}_{\vec{c}}(\vec{x}) \in V_z | \mathbf{x}_4 = \dots = \mathbf{x}_{t+1} = \mathbf{0}] \geq 1 - 1/(2|\mathbb{F}|^{2m})$ for all $\mathbf{z} \in H^m$ (random $\vec{c} \leftarrow V_0$ satisfies these properties with probability at least $1 - |\mathbb{F}|^{-1}$).

In the first step we show that $f_{c_1,1}$ is linear, so $f_{c_1,1}(\mathbf{x}_1) = \mathbf{A}_1 \mathbf{x}_1$ for a matrix $\mathbf{A}_1 \in \mathbb{F}^{m \times m}$; similarly, we have $f_{c_2,2}(\mathbf{x}_2) = \mathbf{A}_2 \mathbf{x}_2$.

In the second step we show that \mathbf{A}_1 and \mathbf{A}_2 are equal (so drop the subscripts). In particular, it follows that f_1 and f_2 are affine with the same linear component: $f_i(\mathbf{x}_i) = f_{c_i,i}(\mathbf{x}_i - \mathbf{c}_i) + f_i(\mathbf{c}_i) = \mathbf{A} \mathbf{x}_i - (f_{c_i,i}(\mathbf{c}_i) - f_i(\mathbf{c}_i))$ for $i = 1, 2$. This argument extends to all $i \in [k]$: there exists $\mathbf{A} \in \mathbb{F}^{m \times m}$ and $\mathbf{b}_1, \dots, \mathbf{b}_k \in \mathbb{F}^m$ such that $\vec{f}(\vec{x}) = (\mathbf{A} \mathbf{x}_1 + \mathbf{b}_1, \dots, \mathbf{A} \mathbf{x}_k + \mathbf{b}_k)$.

Finally, we complete the proof by showing that $\mathbf{A} = \mathbb{I}$ and $\vec{\mathbf{b}} \in V_0$.

To see that $f_{c_1,1}$ is linear, let us first set some notation. Write $(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3) \in V'_0$ if there exists $\vec{x}' \in V_0$ such that $(\mathbf{x}'_1, \mathbf{x}'_2, \mathbf{x}'_3) = (\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3)$ and $\mathbf{x}'_4 = \dots = \mathbf{x}'_{t+1} = \mathbf{0}$. Clearly $(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3), (\mathbf{x}'_1, \mathbf{x}'_2, \mathbf{x}'_3) \in V'_0$ implies $\mathbf{x}'_1 = \mathbf{x}_1$ because of agreement considerations. Note that $V'_0 \subset \mathbb{F}^{3m}$ is a $(2m)$ -dimensional subspace so is closed under addition. By our choice of $\vec{c} \in V_0$, $(f_{c_1,1}(\mathbf{x}_1), f_{c_2,2}(\mathbf{x}_2), f_{c_3,3}(\mathbf{x}_3)) \in V'_0$ whenever $(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3) \in V'_0$. Now, choose arbitrary $\mathbf{x}_1, \mathbf{x}'_1 \in \mathbb{F}^m$, we will show $f_{c_1,1}(\mathbf{x}_1 + \mathbf{x}'_1) = \vec{\mathbf{x}}_1 + \vec{\mathbf{x}}'_1$, where $\vec{\mathbf{x}}_1 = f_{c_1,1}(\mathbf{x}_1)$, $\vec{\mathbf{x}}'_1 = f_{c_1,1}(\mathbf{x}'_1)$. Let $\mathbf{x}_2, \mathbf{x}_3 \in \mathbb{F}^m$ be so that $(\mathbf{x}_1, \mathbf{x}_2, \mathbf{0}), (\mathbf{x}'_1, \mathbf{0}, \mathbf{x}_3) \in V'_0$ (such $\mathbf{x}_2, \mathbf{x}_3$ exist by interpolation). Let $\vec{\mathbf{x}}_i = f_{c_i,i}(\mathbf{x}_i)$ for $i = 2, 3$. We have $(\vec{\mathbf{x}}_1, \vec{\mathbf{x}}_2, \mathbf{0}), (\vec{\mathbf{x}}'_1, \mathbf{0}, \vec{\mathbf{x}}_3) \in V'_0$ (using $\vec{f}_{\vec{c}}(\mathbf{0}) = \mathbf{0}$), and so $(\vec{\mathbf{x}}_1 + \vec{\mathbf{x}}'_1, \vec{\mathbf{x}}_2, \vec{\mathbf{x}}_3) \in V'_0$. On the other hand, if we first add then apply \vec{f} we get $(f_{c_1,1}(\mathbf{x}_1 + \mathbf{x}'_1), \vec{\mathbf{x}}_2, \vec{\mathbf{x}}_3) \in V'_0$; $f_{c_1,1}(\mathbf{x}_1 + \mathbf{x}'_1) = \vec{\mathbf{x}}_1 + \vec{\mathbf{x}}'_1$ follows.

Let $\mathbf{A}_1 \in \mathbb{F}^{m \times m}$ be the matrix form of $f_{c_1,1}$. As mentioned before, the above argument also shows that $f_{c_2,2}$ is linear with matrix $\mathbf{A}_2 \in \mathbb{F}^{m \times m}$. We show here that $\mathbf{A}_1 = \mathbf{A}_2$. To see this, consider the linear map $\Phi : \mathbb{F}^m \rightarrow \mathbb{F}^m$ which sends \mathbf{x}_1 to \mathbf{x}_2 such that $(\mathbf{x}_1, \mathbf{x}_2, \mathbf{0}) \in V'_0$. Note that Φ is actually just multiplication by some non-zero scalar $\beta \in \mathbb{F}$ (this can be seen by writing Φ explicitly in terms of Lagrange interpolation coefficients). For any $\mathbf{x}_1 \in \mathbb{F}^m$, $(\mathbf{x}_1, \beta \cdot \mathbf{x}_1, \mathbf{0}) \in V'_0$; the properties of $\vec{f}_{\vec{c}}$ imply that $(\mathbf{A}_1 \mathbf{x}_1, \beta \cdot \mathbf{A}_2 \mathbf{x}_1, \mathbf{0}) \in V'_0$; this means $\beta \mathbf{A}_2 \mathbf{x}_1 = \beta \mathbf{A}_1 \mathbf{x}_1$; $\mathbf{A}_1 = \mathbf{A}_2$ follows.

We have shown so far that f_1 and f_2 are affine with the same linear component. This argument extends to any f_i, f_j for $i, j \in [k]$ so, as mentioned above, there exists a single matrix $\mathbf{A} \in \mathbb{F}^{m \times m}$ and vectors $\mathbf{b}_1, \dots, \mathbf{b}_k \in \mathbb{F}^m$ such that $f_i(\mathbf{x}_i) = \mathbf{A} \mathbf{x}_i + \mathbf{b}_i$ for all $i \in [k]$. We show that $\mathbf{A} = \mathbb{I}$ and $\vec{\mathbf{b}} \in V_0$, which completes the proof. For any $\mathbf{z} \in H^m$, the affine map $\vec{f} : \mathbb{F}^{mk} \rightarrow \mathbb{F}^{mk}$ maps the affine plane $V_z \subset \mathbb{F}^{mk}$ to an affine plane in \mathbb{F}^{mk} . It follows that either $\vec{f}(V_z) \subset V_z$, or $\Pr_{\vec{x} \leftarrow V_z} [\vec{f}(\vec{x}) \in V_z] \leq |\mathbb{F}|^{-1}$ since affine planes intersect in at most $1/|\mathbb{F}|$ fraction of their points unless there is containment. We are given that $\Pr_{\vec{x} \leftarrow V_z} [\vec{f}(\vec{x}) \in V_z] \geq 1 - |\mathbb{F}|^{-(2m+2)}$, so it must be that $\vec{f}(V_z) \subset V_z$ for all $\mathbf{z} \in H^m$. In particular, $\vec{\mathbf{b}} = \vec{f}(\mathbf{0}) \in V_0$. Finally, note that $\vec{x} \mapsto (\mathbf{A} \mathbf{x}_1, \dots, \mathbf{A} \mathbf{x}_k) + \vec{\mathbf{b}}$ maps V_z to $V_{\mathbf{A}z}$. Therefore, we must have $\mathbf{A}z = z$ for all $\mathbf{z} \in H^m$. As H^m spans \mathbb{F}^m , this forces $\mathbf{A} = \mathbb{I}$ as desired. \square \square

Acknowledgments

We thank Mariana Raykova for motivational discussions in the early stages of this project, and Madhu Sudan and Yael Kalai for breaking some of our schemes.

References

- [Ajt10] Miklós Ajtai. Oblivious rams without cryptographic assumptions. In *STOC*, pages 181–190. ACM, 2010.
- [BI01] Amos Beimel and Yuval Ishai. Information-theoretic private information retrieval: A unified construction. In *ICALP*, volume 2076 of *Lecture Notes in Computer Science*, pages 912–926. Springer, 2001.
- [BIM04] Amos Beimel, Yuval Ishai, and Tal Malkin. Reducing the servers' computation in private information retrieval: PIR with preprocessing. *J. Cryptology*, 17(2):125–151, 2004.

- [BIPW17] Elette Boyle, Yuval Ishai, Rafael Pass, and Mary Wootters. Can we access a database both locally and privately? Cryptology ePrint Archive, Report 2017/567, 2017. <http://eprint.iacr.org/2017/567>.
- [BN00] Daniel Bleichenbacher and Phong Q. Nguyen. Noisy polynomial interpolation and noisy chinese remaindering. In *EUROCRYPT*, volume 1807 of *Lecture Notes in Computer Science*, pages 53–69. Springer, 2000.
- [Bra09] Mark Braverman. Poly-logarithmic independence fools ac^0 circuits. In *IEEE Conference on Computational Complexity*, pages 3–8. IEEE Computer Society, 2009.
- [BTV12] Mihir Bellare, Stefano Tessaro, and Alexander Vardy. Semantic security for the wiretap channel. In *CRYPTO*, volume 7417 of *Lecture Notes in Computer Science*, pages 294–311. Springer, 2012.
- [CKGS98] Benny Chor, Eyal Kushilevitz, Oded Goldreich, and Madhu Sudan. Private information retrieval. *J. ACM*, 45(6):965–981, 1998.
- [CMS99] Christian Cachin, Silvio Micali, and Markus Stadler. Computationally private information retrieval with polylogarithmic communication. In *EUROCRYPT*, volume 1592 of *Lecture Notes in Computer Science*, pages 402–414. Springer, 1999.
- [CS03] Don Coppersmith and Madhu Sudan. Reconstructing curves in three (and higher) dimensional space from noisy data. In *Proceedings of the 35th Annual ACM Symposium on Theory of Computing, June 9-11, 2003, San Diego, CA, USA*, pages 136–142, 2003.
- [CSP⁺16] Raymond Cheng, William Scott, Bryan Parno, Irene Zhang, Arvind Krishnamurthy, and Thomas Anderson. Talek: a private publish-subscribe protocol. 2016.
- [DMN11] Ivan Damgård, Sigurd Meldgaard, and Jesper Buus Nielsen. Perfectly secure oblivious RAM without random oracles. In *TCC*, volume 6597 of *Lecture Notes in Computer Science*, pages 144–163. Springer, 2011.
- [Dod12] Yevgeniy Dodis. Shannon impossibility, revisited. In *ICITS*, volume 7412 of *Lecture Notes in Computer Science*, pages 100–110. Springer, 2012.
- [GKS10] Parikshit Gopalan, Subhash Khot, and Rishi Saket. Hardness of reconstructing multivariate polynomials over finite fields. *SIAM Journal on Computing*, 39(6):2598–2621, 2010.
- [GO96] Oded Goldreich and Rafail Ostrovsky. Software protection and simulation on oblivious rams. *J. ACM*, 43(3):431–473, 1996.
- [GR17] Oded Goldreich and Guy N. Rothblum. Simple doubly-efficient interactive proof systems for locally-characterizable sets. *Electronic Colloquium on Computational Complexity (ECCC)*, 24:18, 2017.
- [KO97] Eyal Kushilevitz and Rafail Ostrovsky. Replication is NOT needed: SINGLE database, computationally-private information retrieval. In *FOCS*, pages 364–373. IEEE Computer Society, 1997.
- [KR17] Yael Kalai and Ran Raz. personal communication. 2017.
- [MRS09] Ben Morris, Phillip Rogaway, and Till Stegers. How to encipher messages on a small domain. In *CRYPTO*, volume 5677 of *Lecture Notes in Computer Science*, pages 286–302. Springer, 2009.
- [NP06] Moni Naor and Benny Pinkas. Oblivious polynomial evaluation. *SIAM J. Comput.*, 35(5):1254–1281, 2006.
- [SCSL11] Elaine Shi, T.-H. Hubert Chan, Emil Stefanov, and Mingfei Li. Oblivious RAM with $o((\log n)^3)$ worst-case cost. In *Advances in Cryptology - ASIACRYPT 2011 - 17th International Conference on the Theory and Application of Cryptology and Information Security, Seoul, South Korea, December 4-8, 2011. Proceedings*, pages 197–214, 2011.