Secret-Key PIR from Random Linear Codes

Caicai Chen [*] Yuval Ishai [†] Tamer Mour [‡] Alon Rosen [§]

April 8, 2025

Abstract

Private information retrieval (PIR) allows to privately read a chosen bit from an N-bit database x with $o(N)$ bits of communication. Lin, Mook, and Wichs (STOC 2023) showed that by preprocessing x into an encoded database $\widehat{x}$, it suffices to access only polylog(N) bits of $\widehat{x}$ per query. This requires $|\widehat{x}| \geq N \cdot \text{polylog}(N)$, and prohibitively large server circuit size.

We consider an alternative preprocessing model (Boyle et al. and Canetti et al., TCC 2017), where the encoding $\widehat{x}$ depends on a client's short secret key. In this *secret-key* PIR (sk-PIR) model we construct a protocol with $O(N^{\varepsilon})$ communication, for any constant $\varepsilon > 0$, from the Learning Parity with Noise assumption in a parameter regime not known to imply public-key encryption. This is evidence against public-key encryption being necessary for sk-PIR.

Under a new conjecture related to the hardness of learning a hidden linear subspace of $\mathbb{F}_2^n$ with noise, we construct sk-PIR with similar communication and encoding size $|\widehat{x}| = (1+\varepsilon) \cdot N$ in which the server is implemented by a Boolean circuit of size $(4+\varepsilon) \cdot N$. This is the first candidate PIR scheme with such a circuit complexity.

[*]Bocconi University. `caicai.chen@unibocconi.it`. Work supported by European Research Council (ERC) under the EU's Horizon 2020 research and innovation programme (Grant agreement No. 101019547).

[†]Technion and AWS. `yuval.ishai@gmail.com`. Supported by ISF grants 3527/24, BSF grant 2022370, and ISF-NSFC grant 3127/23. This paper describes work performed at the Technion and is not associated with Amazon. Part of this work done while visiting Bocconi university, supported by European Research Council (ERC) under the EU's Horizon 2020 research and innovation programme (Grant agreement No. 101019547).

[‡]Bocconi University. `tamer.mour@unibocconi.it`. Work supported by European Research Council (ERC) under the EU's Horizon 2020 research and innovation programme (Grant agreement No. 101019547).

[§]Bocconi University. `alon.rosen@unibocconi.it`. Work supported by European Research Council (ERC) under the EU's Horizon 2020 research and innovation programme (Grant agreement No. 101019547) and Cariplo CRYPTONOMEX grant.

Contents

1 Introduction

Private information retrieval (PIR) is a fundamental building block for sublinear-communication cryptography. It was introduced by Chor, Goldreich, Kushilevitz and Sudan in the multi-server setting [CGKS95] and by Kushilevitz and Ostrovsky [KO97] in the single-server case.

A PIR protocol allows a client to read the i-th bit from a database $x \in \{0,1\}^N$ while computationally hiding i from the server storing x. By this we mean that any poly(N)-time server has only an $N^{-\omega(1)}$ advantage in distinguishing between two distinct client queries i, i'.

Our focus is on single-server 2-round PIR, which involves a single question by the client followed by an answer from the server. Crucially, such a PIR protocol should only use $o(N)$ bits of communication, ruling out the trivial solution of downloading the entire database.

1.1 PIR with preprocessing

In the standard PIR model, the server must read every bit of the database; if x_i is not read, the server knows that x_i was not queried by the client. Motivated by the goal of PIR with sublinear server computation in the RAM model, Beimel, Ishai and Malkin [BIM00] proposed a relaxed model of PIR in which the server can store an encoding $\widehat{x}$ of the database x.

This encoding is computed once, in an offline *preprocessing phase*, and stored on an otherwise-stateless server. Then, in the online phase, a stateless client can make an arbitrary number of PIR queries to the database, where to answer each query the server only needs to read $o(N)$ bits from $\widehat{x}$ and communicate $o(N)$ bits. Such a protocol is referred to as *doubly efficient* PIR.

A recent breakthrough work of Lin, Mook and Wichs (LMW) [LMW23] realized the strongest flavor of doubly efficient PIR, where $\widehat{x}$ is a public (and deterministic) encoding of x, under the standard Ring-LWE assumption [LPR10]. While settling the crude asymptotic question of doubly efficient PIR, the LMW protocol is still impractical. This is due in part to a large polylogarithmic storage overhead. Furthermore, when considering the alternative cost metric of server *circuit size*, the LMW protocol has an even bigger overhead than PIR protocols in the plain model.

1.2 Secret-Key PIR

The first evidence for the feasibility of single-server doubly efficient PIR was actually given in a different preprocessing model, in earlier works by Boyle, Ishai, Pass and Wootters [BIPW17] and Canetti, Holmgren and Richelson [CHR17]. Their main result was a candidate construction of a PIR protocol with *secret-key* preprocessing (sk-PIR), in which the encoding $\widehat{x}$ is generated from x using a short secret key which is known to the client but not to the server.

The sk-PIR protocols from [BIPW17, CHR17] encode the database using a secretly permuted Reed-Muller (RM) code, and can achieve polylog(N) communication and server computation in the RAM model. Security relies on an ad-hoc and "highly structured" assumption. In spite of subsequent analysis [BHW19, BW21, BHMW21], this assumption is still quite poorly understood.

We take a step back and consider the bare sk-PIR model, requiring only sublinear communication without insisting on the standard notion of double efficiency. We ask the following questions:

- **Feasibility.** In which "cryptographic world" does secret-key PIR live?
- **Efficiency.** What is the *circuit complexity* of sk-PIR, namely the minimal size of a Boolean circuit required to compute the server's answer from $\widehat{x}$ and the client's question?

In the context of feasibility, standard PIR protocols, as well as PIR with public preprocessing, imply 2-round oblivious transfer [DMO00] which in turn implies public-key encryption (PKE). They also imply collision-resistant hashing (CRH) [IKO05]. For sk-PIR, it is only known that one-way functions are necessary [BIPW17]. Are PKE or CRH also necessary?

With respect to efficiency, in all existing PIR protocols, including ones based on strong assumptions or even purely heuristic ones, the server's circuit size is at least $N \cdot \text{polylog}(N)$. Such protocols have a polylogarithmic computational overhead (in the Boolean circuit model) compared to the insecure baseline of computing the selection function that maps (x, i) to x_i. This should be contrasted with the related primitives of oblivious transfer [IKOS08, BCG+23] and CRH [AHI+17], for which *constant computational overhead* was achieved under plausible assumptions.

1.3 Our Results

Our starting point is a natural attempt for modifying the sk-PIR blueprint from [BIPW17, CHR17]. Suppose that instead of using a secretly permuted RM code, which is a specific distribution over locally decodable codes, the client encodes the database using a *random linear code*. What are the feasibility and efficiency consequences?

It turns out that this simple idea, in conjunction with noisy queries by the client, yields surprisingly powerful consequences:

- Secret-key PIR with $O(N^{\varepsilon})$ communication, for any constant $\varepsilon > 0$, from the Learning Parity with Noise (LPN) assumption in a parameter regime not known to imply PKE or CRH, giving evidence against proving PKE is necessary for attaining sk-PIR.
- Secret-key PIR with similar communication and encoding size $|\widehat{x}| = (1 + \varepsilon) \cdot N$ in which the server is implemented by a Boolean circuit of size $(4 + \varepsilon) \cdot N$, assuming a new *Learning Subspace with Noise* (LSN) conjecture on learning hidden linear subspaces of $\mathbb{F}_2^n$ with noise.

The new class of "Learning Subspace with Noise" assumptions considers the pseudorandomness of noisy samples from a secret linear code. Our LSN assumptions can be viewed as less structured variants of the assumption underlying the previous sk-PIR from [BIPW17, CHR17]. They are also closely related to the problem of learning *mixtures* of uniform distributions over linear subspaces, studied by Chen, De, and Vijayaraghavan [CDV21]. The study of these LSN assumptions is of independent theoretical interest.

The strongest version of our second result gives a PIR scheme with far better circuit complexity than all alternative approaches we are aware of. This requires assuming hardness of what we call the *split-LSN* problem, a more structured variant of our basic LSN assumption.

Our LSN-based constructions have several other useful features. They can support slightly sublinear server computation even with a very small storage overhead, which does not seem possible via the lattice-based approach from [LMW23]. They also seem attractive for practical implementation, offering concrete efficiency advantages over competing approaches. Finally, they can be applied beyond the designated-client setting by either using general-purpose obfuscation techniques or, more realistically, by distributing the role of the client between two or more parties (say, the client and the server) using secure multiparty computation. The latter distributed variant of sk-PIR may be almost as good as the public variant for applications that inherently require a non-collusion assumption, which is commonly the case in threshold cryptography.

In the coming subsections we provide more details on each of the above results, as well as on the underlying new assumptions and various optimizations that we employ.

1.4 Secret-Key PIR from Learning Parity with Noise

On the feasibility front, we show that using a secret random linear code enables sk-PIR from the standard Learning Parity with Noise (LPN) assumption [BFKL94] in a parameter regime not known to imply PKE or CRH, let alone PIR. Recall that LPN asserts the pseudorandomness of a noisy random codeword in a (public) random k-dimensional linear code over $\mathbb{F}_2$ (see Definition 3.4).

Theorem 1.1 (sk-PIR from LPN, Informal)**.** *For every $\gamma > 0$ and $\varepsilon > 0$, LPN with noise rate $1/k^\gamma$ implies sk-PIR with communication $O(N^\varepsilon)$ and storage $|\widehat{x}| = \mathrm{poly}(N)$. Furthermore, for sufficiently large $\gamma = \gamma(\varepsilon) < 1$, the storage can be improved to $O(N^{1+\varepsilon})$.*

For $\gamma < 1/2$, constructing PKE from LPN with noise rate $1/k^\gamma$ is a major open problem. In light of this, Theorem 1.1 can be viewed as a strong barrier to proving that sk-PIR implies PKE, suggesting that it may live in an intermediate world between "Minicrypt" and "Cryptomania." Even for $1/2 < \gamma < 1$, where LPN with noise rate $1/k^\gamma$ is known to imply PKE [Ale03], it is not known to imply CRH, let alone PIR in the plain model.[1]

1.5 Better Efficiency via Learning Subspace with Noise

In the following, we let $\mathbb{F}$ denote a finite field. While we state our assumptions for a general $\mathbb{F}$, all of our protocols can use $\mathbb{F} = \mathbb{F}_2$. One may therefore restrict the attention to this case.

An LSN problem concerns the task of distinguishing between polynomially many random codewords $\mathbf{c}_i$ sampled from a secret random linear code $C \subseteq \mathbb{F}^n$ and uniformly random vectors, when the random codewords are subject to noise. The LSN instances are parameterized by the *dimension* $k := k(\lambda) \geq \lambda$ of the random code and its *block length* $n := n(\lambda) > k$. Here and elsewhere, λ is a cryptographic security parameter; by $X_\lambda \approx_c Y_\lambda$ we mean that any $\mathrm{poly}(\lambda)$-time algorithm has a $\lambda^{-\omega(1)}$ advantage in distinguishing between X_λ and Y_λ.

We will consider two types of noise: *mixture noise*, replacing each $\mathbf{c}_i$ with probability $\mu := \mu(\lambda)$ by a sample from a different distribution (a uniform vector in $\mathbb{F}^n$ by default), or *planting noise*, hiding $\mathbf{c}_i$ in a random low-dimensional linear or affine space containing it. We start with our default version of LSN that uses a simple mixture noise.

Definition 1.1 (Basic LSN)**.** *The* learning subspace with noise *assumption* (k, n, μ)-LSN *asserts that for a uniformly random secret rank-k matrix $\mathbf{C} \in \mathbb{F}^{k\times n}$ and any polynomial number of samples $m := m(\lambda)$, it holds that:*

$$(\mathbf{c}_1 + \mathbf{e}_1, \ldots, \mathbf{c}_m + \mathbf{e}_m) \approx_c (\mathbf{u}_1, \ldots, \mathbf{u}_m)\,,$$

where $\mathbf{c}_i = \mathbf{a}_i^\intercal \mathbf{C}$ for $\mathbf{a}_i \leftarrow \mathbb{F}^k$, $\mathbf{e}_i$ is uniformly random in $\mathbb{F}^n$ with probability μ and $\mathbf{e}_i = \mathbf{0} \in \mathbb{F}^n$ otherwise, and $\mathbf{u}_i \leftarrow \mathbb{F}^n$.

For $n = k + 1$, the *search version* of the LSN assumption over $\mathbb{F}_2$ was shown to be equivalent to the standard LPN assumption with noise rate $\mu/2$ [CDV21]. We observe that this also holds for the decision version, and obtain several other useful results on LSN and its relation with LPN.

Theorem 1.2 (LSN facts, Informal)**.** *The following holds for (k, n, μ)-LSN over $\mathbb{F}_2$.*

1. **LSN distinguisher.** *If $\mu < 1 - (k/n)^d$, there is an $n^{O(d)}$-time LSN distinguisher.*

[1]LPN with noise rate $\log^2 k/k$ implies CRH [BLVW19, YZW+19]. LPN with an even lower noise rate of $\log^{1+\beta} k/k$, where $0 < \beta < 1$, is known to imply (plain-model) PIR with slightly sublinear communication of $N/2^{\Theta(\log^{1-\beta} N)}$ [AMR25]. Obtaining fully sublinear PIR from any flavor of LPN is open.

2. **LSN implies LPN**. *For constant code rate $\rho = k/n$ and $\eta = 1 - \mu = o(1)$, (k, n, μ)-LSN implies LPN with code dimension k, code length $k \cdot (1 + \Omega(\eta))$, and noise rate η.*

3. **LPN implies "sparse-noise" LSN**. *LPN with noise rate ε implies a variant of LSN with the following noise pattern: Let I be a random set of k linearly independent columns of $\mathbf{C}$. Then, for each sampled codeword $\mathbf{c}_i$, we flip each bit* outside *I with ε probability.*

See Section 2 for proof overview. The distinguisher from part (1) will guide our parameter choice for LSN. Part (3) serves as a basis for the LPN-based construction of Theorem 1.1, which follows as a special case of an sk-PIR construction based on LSN with general noise.

We put forward the following conjecture.

Conjecture 1.1 (LSN conjecture). *For every $0 < \rho < 1$, the (k, n, μ)-LSN assumption holds when $k \geq \rho n$ and $\mu \geq 1 - o(1)$.*

Under the LSN conjecture, we can make the storage nearly optimal, or alternatively get a weak notion of "doubly efficient" PIR.

Theorem 1.3 (Low-storage or sublinear-computation sk-PIR from LSN, Informal). *Suppose Conjecture 1.1 holds. Then, for every $\varepsilon > 0$, there is sk-PIR with communication $O(N^{\varepsilon})$ and storage $|\widehat{x}| = (1+\varepsilon) \cdot N$. Alternatively, with $N^{1+\varepsilon}$ storage, the server only reads $N/\mathrm{polylog}(N)$ bits from $\widehat{x}$.*

While the latter can be viewed as a very weak form of doubly efficient sk-PIR, it is still surprising that this can be achieved using *random* linear codes. Indeed, the database encoding in any doubly efficient sk-PIR protocol must support (smooth) local decoding with sublinear query complexity [BIPW17]. This is achieved here via concatenation with the Hadamard code, in the spirit of a technique used in [BIM00] to realize *multi-server* PIR with similar server efficiency.

1.6 Minimizing Server Complexity via Splitting

Motivated by the goal of minimizing the computational complexity of PIR in the Boolean circuit model, we consider a "split" variant of LSN that plants each codeword $\mathbf{c}_i$ in a low-dimensional affine space[2] $\mathbf{c}_i + V_i$, where V_i is a product of s linear spaces of dimension n/s. This can be viewed as splitting the coordinates of $\mathbf{c}_i$ into s blocks, and hiding each block in an affine space.

Definition 1.2 (Split-LSN). *The* split learning subspace with noise *assumption* (k, n, r, s)-SLSN *asserts that for a uniformly random secret rank-k matrix $\mathbf{C} \in \mathbb{F}^{k \times n}$ and any polynomial number of samples $m := m(\lambda)$, it holds that:*

$$((\mathbf{V}_{i,1}, \mathbf{c}_{i,1} + \mathbf{e}_{i,1}), \ldots, (\mathbf{V}_{i,s}, \mathbf{c}_{i,s} + \mathbf{e}_{i,s}))_{i \in [m]} \approx_c ((\mathbf{V}_{i,1}, \mathbf{u}_{i,1}), \ldots, (\mathbf{V}_{i,s}, \mathbf{u}_{i,s}))_{i \in [m]},$$

where $\mathbf{c}_i = \mathbf{a}_i^{\intercal} \mathbf{C}$ for $\mathbf{a}_i \leftarrow \mathbb{F}^k$, $(\mathbf{c}_{i,1}, \ldots, \mathbf{c}_{i,s})$ is a partitioning of $\mathbf{c}_i$ into blocks of length n/s, and for any i, j, $\mathbf{u}_{i,j} \leftarrow \mathbb{F}^{n/s}$ and $\mathbf{e}_{i,j} = \mathbf{d}_{i,j}^{\intercal} \mathbf{V}_{i,j}$ for $\mathbf{V}_{i,j} \leftarrow \mathbb{F}^{(r-1) \times (n/s)}$ and $\mathbf{d}_{i,j} \leftarrow \mathbb{F}^{r-1}$.

With $s = 1$, split-LSN is (at least) as secure as a "regular" variant of LSN with noise rate $\mu = 1 - 1/r$ in which each chunk of r samples contains exactly one codeword and $r - 1$ noise vectors. A similar regular variant of LPN has been extensively studied in the literature; see [AG11, LWYY24] and references therein. However, such a regular variant of LSN would require r to be super-constant,

[2]One could alternatively plant each block in a random *linear space* containing it; however, this would require a slightly more complicated version of the split-LSN assumption that allows for shifting before planting.

which conflicts with our efficiency goals. Splitting allows us to have noise with super-constant entropy while keeping r constant by accumulating the entropy across blocks.

With $s \gg n/k$, split-LSN may be loosely viewed as a less structured (and seemingly more conservative) variant of the assumption underlying the sk-PIR protocol proposed in [BIPW17, CHR17] and further analyzed in [BHW19, BW21, BHMW21]. In this protocol, the secret code C is a (dual) RM code, obtained by applying a random secret permutation to the coordinates of a fixed RM code. Each sample is obtained by picking a random *low-weight* codeword of C and revealing only its support set, namely the locations of the nonzero entries but not their values. (This set includes a random subset of points on a permuted low-degree curve.) Since the codeword has low weight, each of the s parts includes only one nonzero symbol with substantial probability, thus we can restrict the LSN samples to such instances. Revealing only the location of the nonzero symbol can be viewed as hiding it in a 1-dimensional linear space. While this is by no means a formal security reduction, future insights on split-LSN seem likely to imply insights on the permuted RM assumption and vice versa.

The split-LSN assumption over $\mathbb{F}_2$ with $r = 2$ will help us minimize the circuit complexity of the server. For this choice of parameters, hiding each block $\mathbf{c}_{i,j}$ in an affine space of dimension $r-1$ simply means that we reveal a pair of vectors containing $\mathbf{c}_{i,j}$ and a random "distractor" $\mathbf{r}_{i,j} \in \mathbb{F}_2^{n/s}$ in a random order. We conjecture that it suffices for the splitting parameter s to be such that the length of each block is sufficiently smaller than the code dimension k, and have empirical evidence supporting this conjecture with respect to a class of algebraic attacks.

Conjecture 1.2 (Split-LSN conjecture). *For every $\delta > 0$ and $0 < \rho < 1$, the $(k, n, 2, s)$-SLSN assumption holds when $k \geq \rho n$ and $s \geq (n/k) \cdot n^{\delta}$.*

Compared to basic LSN with constant code rate $0 < \rho < 1$ and noise rate $\mu = 1 - n^{-\delta}$, the above split-LSN conjecture seems more aggressive in that the noise is more structured, but more conservative in that low-degree distinguishers seem less effective. In particular, while part (1) of Theorem 1.2 implies a quasi-polynomial time distinguisher for LSN with these parameters, we are not aware of such a distinguisher for the split-LSN variant. See Section 4.2 for discussion.

Under the split-LSN conjecture, we can bring the Boolean circuit size of the server (over the B_2 basis, consisting of gates $g : \{0,1\}^2 \to \{0,1\}$) to roughly $4N$, which is optimal up to a small multiplicative constant.

Theorem 1.4 (Server-efficient sk-PIR from split-LSN, Informal). *Suppose Conjecture 1.2 holds over $\mathbb{F} = \mathbb{F}_2$. Then, for every $\varepsilon > 0$, there is sk-PIR with communication $O(N^{\varepsilon})$ and storage $(1+\varepsilon) \cdot N$, where the server can be implemented by a Boolean circuit of size $(4+\varepsilon) \cdot N$.*

An alternative candidate for sk-PIR with linear circuit complexity can be based on the permuted Reed-Muller code protocol from [BIPW17, CHR17]. With a careful choice of parameters, this protocol can be instantiated with $|\widehat{x}| = O(N)$, where the server reads $O(N^{\varepsilon})$ bits from $\widehat{x}$, for any $\varepsilon > 0$. Combining this with a recent construction of linear-size circuits for multiselection [HR24], the server can be implemented by a Boolean circuit of size $O(N)$, but with a very large hidden constant. Here the constant is close to optimal and the assumption is less structured.

We are not aware of any other (single-server) PIR schemes that have linear circuit complexity. In all lattice-based PIR schemes, both theoretical (e.g., [LMW23]) and applied (e.g., [HHC+23]), the server circuit complexity is at least $N \cdot \text{polylog}(N)$. Number-theoretic PIR schemes (e.g., [KO97]) have an even higher computational cost.

Under a variant of Conjecture 1.2, with rate $\rho = o(1)$ where hardness of split-LSN is assumed also to hold for *quasi-polynomial* algorithms, the result from Theorem 1.4 can be improved to sk-PIR with $N^{o(1)}$ communication, where the server has probe complexity $o(N)$ and otherwise the same storage and circuit size. Such a protocol is "doubly efficient" in a weak sense similar to Theorem 1.3, and has the advantage of small storage size. It is not clear how to apply the lattice-based approach from [LMW23] to achieve sublinear server computation in this regime.

Concrete efficiency. To give a sense of potential practical utility: In the above sk-PIR protocol, the amortized server work for processing 64 bits of the database involves one XOR operations of 64-bit strings. For comparison, a typical instantiation of the SimplePIR protocol [HHC+23] (which uses *public* preprocessing) requires 8 multiplications and 8 additions over $\mathbb{Z}_{2^{32}}$. Thus, the number of *bit operations* performed by our sk-PIR protocol is smaller by roughly 2 orders of magnitude. This advantage is not expected to fully materialize on standard CPU hardware, due to the cost of memory access and native support of $\mathbb{Z}_{2^{32}}$ operations. However, we believe that, with a suitable architecture, our approach can lead to a significant practical speedup compared to existing single-server PIR protocols. This may further motivate future analysis of split-LSN and related assumptions.

1.7 Minimizing Client Complexity

Our focus so far was on communication and server computation of sk-PIR, ignoring the client computation. In all of our protocols that have $O(N^{\varepsilon})$ communication, the client can be implemented in (slightly) sublinear time $\tilde{O}(N^{1-\varepsilon})$. We can further reduce the client's runtime by using two different approaches:

- By composing a simple variant of our protocols that has $O(N^{1/2})$ communication with a standard PIR protocol that has $O(N^{\varepsilon})$ communication and client computation, we inherit the server complexity of our protocols and the client complexity of the standard protocol. The existence of such standard PIR protocols is arguably a mild assumption in the context of our efficiency goals, where we are willing to make new assumptions to derive new conclusions.
- Based on LPN alone, and building on recent techniques from [BN25, VZ25], we can simultaneously obtain $O(N^{1/3+\varepsilon})$ communication and client computation.

1.8 Related Work

Alternative models. There is a large body of works studying alternative models for PIR with preprocessing, including Oblivious RAM (ORAM) [GO96] and other PIR models [CK20, ZLTS23]. In all of these alternative models, the server and/or the client need to maintain and update a database-dependent state. In contrast, in sk-PIR both the client and the server are stateless: The client only needs to store a short (and database-independent) key, and the server only needs to store the encoded database. The statelessness feature natively supports concurrent executions of sk-PIR on the same encoded database, e.g., by multiple devices owned by the same organization or multiple secure computation protocols emulating the same (designated) client.

Another (related) difference is that in contrast to sk-PIR, which is known to imply one-way functions [BIPW17], most of the alternative models admit information-theoretic solutions with sublinear communication complexity [Ajt10, DMN11, ISW24].

Finally, sk-PIR can be compared to *searchable symmetric encryption* (SSE) [SWP00], for which one-way functions are sufficient. However, in all known SSE solutions, multiple queries inherently leak the access pattern. Our results show that LPN-style assumptions are sufficient to eliminate this leakage, albeit with a much higher server computation cost in the RAM model.

Assumptions for doubly-efficient sk-PIR. While our main focus in this work is on minimizing the communication and the *circuit size* of the server, some of our constructions also achieve slightly sublinear runtime in the RAM or cell-probe model. For this (standard) notion of doubly-efficient sk-PIR, recent work by Lin, Mook, and Wichs [LMW25] shows that a *black-box* use of standard cryptographic primitives or generic models does not suffice, unless a black-box use of a one-way function is also sufficient. The same work also provides partial evidence against the latter. Our relevant protocols do not circumvent these impossibilities, as they are based on concrete assumptions (similarly to concrete assumptions used in prior related works [BIPW17, CHR17, LMW23]). The results of [LMW25] suggest that a more generic approach may be impossible.

The computational overhead of cryptography. This work is partially motivated by the goal of minimizing the *computational overhead* of sk-PIR in the Boolean circuit model, namely the asymptotic ratio between the size of a secure implementation and an insecure implementation. For a variety of other cryptographic primitives, including pseudorandom generators, oblivious transfer and zero-knowledge proofs, this question has been the topic of a large body of work; see [IKOS08, AM13, dCHI$^+$22, RR22, FLY22, BCG$^+$23] and references therein. Most relevant to the current work, fully homomorphic encryption can be realized with polylogarithmic computational overhead under standard lattice assumptions [GHS12], but realizing it with a constant overhead is a challenging open problem. The problem is open even for the easier task of single-server PIR in the plain model, where no (provable or heuristic) candidate construction is known. In contrast, for the related tasks of oblivious transfer [IKOS08] or collision-resistant hashing [AHI$^+$17], constant-overhead constructions can be based on plausible assumptions.

2 Technical Overview

We start by describing a simple variant of our main protocol where the client's query is long but the answer is short, and where security relies on the basic LSN assumption.

Retrieving the i-th entry of $x \in \mathbb{F}^N$ can be viewed as applying the linear function $x \mapsto \mathbf{w}_i^\intercal x$ where $\mathbf{w}_i$ is the i-th unit vector in $\mathbb{F}^N$. To avoid disclosing $\mathbf{w}_i$, the client could use a random systematic linear code $\mathbf{D} : \mathbb{F}^N \to \mathbb{F}^M$ to encode x, where $\mathbf{D}$ is determined by the client's secret key, and have the server only store the encoding $\widehat{x} = \mathbf{D}(x)$.

The client may now retrieve x_i by sending $\mathbf{q} = \mathbf{c} + \mathbf{w}_i \in \mathbb{F}^M$, where $\mathbf{c}$ is a random vector in the dual code $\mathbf{C} = \mathbf{D}^\perp$. Indeed, it holds that $\mathbf{q}^\intercal \widehat{x} = \mathbf{c}^\intercal \mathbf{D}(x) + \mathbf{w}_i^\intercal \mathbf{D}(x) = 0 + x_i$.

While this approach is sufficient to make each individual query $\mathbf{q}$ random, it still reveals to the server joint information about multiple queries. For instance, if the client reads a sequence of identical indices $(i_1, \ldots, i_m)$, the corresponding queries $(\mathbf{q}_1, \ldots, \mathbf{q}_m)$ will always span a linear subspace of dimension $\leq M - N + 1$, whereas for random indices i_j they will typically span the full space $\mathbb{F}^M$ when $m > N$. To eliminate this leakage, the client adds noise to each $\mathbf{q}_i$ in a way that still enables decoding of x_i. Different types of noise correspond to different flavors of LSN.

A remaining issue is that the server may obtain additional information about the queries from the encoded database. To address this, we let the client hide $\mathbf{D}(x)$ by using a secret random mask $\mathbf{r} \leftarrow \mathbb{F}^M$, and undo the masking when obtaining the answer. An sk-PIR scheme as above with simple mixture noise is described in Fig. 1. The scheme inherits the error probability μ from the noise, which can be eliminated either by repetition, by using a suitable erasure code, or by switching to planting noise as in split-LSN. Finally, the client's secret state can be compressed to a short secret key by using psuedorandomness.

Client Secret Key:

- A uniformly random systematic linear code $\mathbf{D} : \mathbb{F}^N \to \mathbb{F}^M$, defining a dual code $\mathbf{C} \subseteq \mathbb{F}^M$
- A uniformly random mask $\mathbf{r} \leftarrow \mathbb{F}^M$.

Database Encoding:

$C \to S$: on input database $x \in \mathbb{F}^N$, upload $\widehat{x} = \mathbf{D}(x) + \mathbf{r} \in \mathbb{F}^M$.

Query:

$C \to S$: on input $i \in [N]$, send $\mathbf{q} = \mathbf{v} + \mathbf{w}_i \in \mathbb{F}^M$, where

$$\begin{cases} \mathbf{v} \leftarrow \mathbf{C} & \text{is a random vector in the dual code,} \quad w.p.\ 1-\mu \\ \mathbf{v} \leftarrow \mathbb{F}^M & \text{is a uniformly random vector,} \quad w.p.\ \mu \end{cases}$$

$S \to C$: send $a = \mathbf{q}^\intercal \widehat{x} \in \mathbb{F}$

C : use $\mathbf{r} \in \mathbb{F}^M$ to compute $x_i = a + \mathbf{q}^\intercal \mathbf{r} \in \mathbb{F}$

Figure 1: Noisy sk-PIR protocol with short answers.

2.1 Balancing Communication

A first step towards realizing efficient sk-PIR protocols based on the above outline follows a generic *balancing* technique, commonly used in PIR protocols [CGKS95, KO97].[3]

Balancing views the database as a collection of smaller databases, arranged as the rows of a $\sqrt{N} \times \sqrt{N}$ matrix, and applies the protocol over each of the rows to retrieve an entire column that contains the client's requested database bit. In applying this to the protocol from Fig. 1, the client uses the same random code $\mathbf{D}$ to encode all rows in the database matrix. This allows using the same query across all rows. On the other hand, it is important that the client use independently sampled random masks to hide the different database rows, essentially masking the database matrix by a uniformly random mask $\mathbf{R} \leftarrow \mathbb{F}^{\sqrt{N} \times \sqrt{N}}$. (The security of the protocol completely breaks if the client uses the same mask for all rows.) Since the client's query is generated now for a $\sqrt{N}$-size database, this reduces the question length at the expense of a longer answer from the server, which contains one field element for each of the $\sqrt{N}$ elements in the retrieved column. As a result, we obtain a protocol where communication grows with $\sqrt{N}$.

2.2 Further Reducing Communication via Folding

In standard PIR, the balancing technique can be further pushed to convert any protocol where the server's answer is particularly small (e.g. when the protocol has "download rate 1") into a protocol where communication depends on N only polylogarithmically [KO97, IP07].

The idea at a high level is again very simple. Recall that in the balanced protocol from above, the client obtains the server responses corresponding to a column of elements, namely a "column of responses". We view this column as a new database, where the entries are server responses, and let the client apply a new PIR query over the column to retrieve a single response. Assuming the server responses in the base PIR are sufficiently short, say consist of a single bit, the answers in the new "folded" protocol consist of a single bit as well. Thus, we may recursively fold the protocol

[3]While balancing works generically in standard PIR, it is not the case in sk-PIR. The reason is that security in sk-PIR does not generally extend to the the setting where multiple databases are encoded under the same secret key (in which case a client query can be re-used across multiple databases). Balancing is possible in our case since our base protocol can be made to satisfy this property.

over and over again, until we essentially perform PIR queries over very small databases (potentially of constant size) that come from a "hierarchy of folded databases" of logarithmic depth.

The generic outline from [IP07] fails when applied to our setting. The reason is that sk-PIR assumes the database is encoded using the client's secret-key. The column over which we want to recurse consists of arbitrary server responses to sk-PIR queries corresponding to different rows of the original database. Generally, these responses do not presume the structure of an encoded database and it is not clear how to make them so without first communicating them to the client.

The linearity of the database encoding in our base protocol helps us get around this difficulty. We observe that if we carry out the above outline with our base scheme, it is possible to encode the original database columns in a way that results a column of server responses that looks like an encoding of a new database!

Recall that, in our base protocol, we encode the database using a random linear code $\mathbf{D}$ and the client receives a linear combination of codeword locations. In the above balancing, then, this corresponds to applying $\mathbf{D}$ over each of the rows in the database and sending the client a linear combination of the columns of the matrix (since we use the same query for all rows). We propose to additionally apply another random code $\mathbf{D}'$ over the *columns* of the database, in which case the client will receive a linear combination of codewords in $\mathbf{D}'$. Due to linearity, such a response is itself a codeword in $\mathbf{D}'$ – namely an encoded database.

This allows to apply another PIR query over the obtained column. The idea can be carried out recursively to reduce communication further on. By folding the database any constant number of times, we obtain communication of N^{ε} for any arbitrarily small constant $\varepsilon > 0$. We can even apply the folding $\omega(1)$ times to obtain communication $N^{o(1)}$. In such a case, however, the databases at the base of the recursion will be of size $N^{o(1)}$ and, since we think of the database size as the security parameter, we rely on the quasi-polynomial hardness of the underlying LSN problem for security.

Applying linear codes along different dimensions of high-dimensional data corresponds essentially to applying a *tensor code*. Thus, the folded protocol may be more easily viewed as a version of our base protocol where we apply a random linear tensor code to our data and the queries are tensors of corresponding codewords from the dual code. The order of the tensor corresponds to the depth of the recursion. Thus, the higher the order, the lower the communication of the scheme.

Standard balancing can be further applied over a folded protocol, that is, by applying the folding over each row of a matrix database, to obtain a slightly better communication and client computation (we elaborate further on client complexity below).

2.3 Analysis of LSN

We now outline the proofs of the three LSN-related facts stated in Theorem 1.2.

LSN distinguisher via a tensored-rank attack. We show that, if $\mu < 1 - (k/n)^d$, there is an $n^{O(d)}$-time LSN distinguisher. The distinguisher is a simpler variant of an algorithm from [CDV21] for the search version of LSN. The basic observation is that if $\mu < 1 - k/n$, then with $n - 1$ samples we expect to observe linearly dependent samples from C in the pseudorandom experiment but not in the random experiment. By taking the d-tensor of each sample, the relative dimension is decreased from k/n to $(k/n)^d$ without changing the noise rate. Thus, the same rank attack will succeed when $\mu < 1 - (k/n)^d$. The runtime, however, needs to scale polynomially with n^d, the d-tensor dimension. See Proposition 4.2 for details.

LSN implies LPN via syndrome decoding. By combining the duality between LPN and syndrome decoding with a known search-to-decision reduction for LPN [AIK07], we can conclude that if LPN fails, then we have an efficient syndrome decoder that recovers with overwhelming probability a random sparse error vector from its syndrome. This syndrome decoder can be used in the LSN experiment to recover the first (sparse) linear dependence between samples from the hidden code C with inverse-polynomial success probability, which suffices for breaking LSN with inverse-polynomial advantage. Note that unlike the previous (unconditional) distinguisher, which uses a much larger number of samples than required by an unbounded distinguisher, the LPN-based distinguisher has an almost minimal sample complexity. See Proposition 4.1 for details.

LPN implies "sparse-noise" LSN. Recall that the sparse-noise variant of LPN modifies the noise distribution of basic LSN in the following way. Instead of replacing a sampled codeword $\mathbf{c}_i \in C$ by a random vector with probability μ, sparse-noise LSN applies independent ε-Bernoulli noise to a subset of the coordinates of $\mathbf{c}_i$. More concretely, the code C is chosen jointly with a random set $I \subseteq [n]$ of k linearly independent coordinates, and each coordinate *outside* I is flipped with ε probability. Using a systematic generator matrix of C that contains the identity matrix in coordinates I, the above sampling experiment is equivalent to a matrix variant of LPN in which the secret vector is replaced by a secret matrix. This matrix LPN variant is in turn equivalent to standard LPN with the same noise rate [Döt14]. See Proposition 4.3 for more details.

2.4 Replacing LSN by LPN

Our LPN-based protocols are obtained by instantiating our general LSN-based template with the sparse-noise variant of LSN discussed above. This can be done efficiently when k is close to n, say $n = k + k^{1/3}$. In this parameter regime, if we pick the LPN noise rate to be $\varepsilon = k^{-\eta}$ for $\eta > 1/3$, the sparse-noise LSN experiment with the same noise probability will keep the codeword $\mathbf{c}_i$ unchanged except with vanishing probability. Whenever $\mathbf{c}_i$ is unchanged, the client can successfully decode the retrieved item. Note that because the database encoding uses the dual code, the storage overhead with the above parameters is cubic. With $n = k + k^{9/10}$ and $\eta > 9/10$ (a stronger LPN assumption), the storage is $O(N^{10/9})$.

2.5 Minimizing Server Circuit Complexity

Our split-LSN Conjecture (Conjecture 1.2) is specifically tailored for minimizing the server computation in the Boolean circuit model. Recall that in the corresponding LSN experiment, each sampled codeword $\mathbf{c}_i \in \mathbb{F}_2^n$ is split into s blocks, and each block is shuffled with a random distractor in $\mathbb{F}_2^{n/s}$. Considering the base protocol for simplicity, the server multiplies each n/s columns of the database matrix X by the two corresponding blocks. The client, who knows the location of the correct block, can decode the correct matrix-vector product. The overall server computation consists of twice the cost of computing matrix-vector product over $\mathbb{F}_2$ in the clear. Choosing the parameters so that X has $\approx N$ entries, the total circuit size is $\approx 4N$.

2.6 Minimizing Client Computation

With the techniques discussed so far, we are able to obtain sk-PIR protocols with small communication and, under split-LSN, a particularly efficient server. The client computation in our protocols consists of two main components:

1. First, to generate a query, the client must sample a uniformly random codeword from a dual code $\mathbf{C} \subset \mathbb{F}^n$ of dimension k (or few such codewords in the folded protocol). In general, this takes time $O(nk) = O(n^2)$. In the folded protocol, the length of the LSN code at every level

of the folding (or along every dimension of the tensor code) is $M^{1/t}$, where M is the length of an encoded database row and t is the number of folding operations (or the order of the tensor). Therefore, query generation takes $O(M^{2/t})$.

2. Second, upon receiving the server's response, the client must "uncompute" the mask $\mathbf{r} \in \mathbb{F}^M$ by multiplying it with the query vector (see Fig. 1). (Note the client needs to do this only for the row including the target coordinate.) The runtime complexity of such a multiplication is M, which is $O(N^{1-\varepsilon})$ in a balanced protocol with communication $O(N^{\varepsilon})$ (note communication of a balanced protocol, folded or not, is at least the number of rows in the database, i.e. N/M).

To summarize the above, the client complexity in a balanced (unfolded) base protocol is dominated by query generation, which takes time quadratic in a database row, therefore linear in the size of the database. In the folded protocols where $t > 1$, query generation is reduced to $O(N^{\varepsilon})$ and the bottleneck in client computation becomes in dealing with the mask, which takes a slightly sublinear time of $O(N^{1-\varepsilon})$. We take two different approaches to reduce client runtime even further.

Composing with PIR. The first approach is quite generic, and "folds" the sk-PIR with standard PIR where the client can be very efficient. More concretely, the sk-PIR is used to transform the database of size N to a database of slightly sublinear size $N^{1-\varepsilon}$: Arrange the database in $N^{1-\varepsilon}$ rows of size N^{ε} each and apply a sk-PIR query over each of the rows to obtain a sublinear-size column of responses. Then, PIR is used over the sublinear-size column to retrieve the target element. This outline gives communication $O(N^{\varepsilon})$ already when instantiating it with the simple base sk-PIR protocol. Further, as long as the server complexity in the PIR over a sublinear-size database is sublinear, we inherit the server efficiency from the sk-PIR. The client now generates an sk-PIR query for a small N^{ε}-size database and a PIR query over the larger $N^{1-\varepsilon}$-size database, which takes time N^{γ} for arbitrarily small $\gamma > 0$ in known PIR protocols under many standard assumptions (e.g. DDH, QRA and LWE) [KO97, OS07].

LPN-based Optimizations. While composing with PIR reduces the client runtime to be an arbitrarily small polynomial, it requires assuming the existence of PIR with efficient client, which is not known to be attainable based on LPN. To obtain an sk-PIR with efficient client based on LPN alone, we instead rely on different techniques taken from the literature of LPN-based cryptography.

First, we invoke a tool developed in two recent works [BN25, VZ25] to reduce the complexity of uncomputing the mask. These works show, based on LPN, how to sample a pseudorandom "trapdoored matrix" $\mathbf{M}$ together with a trapdoor, such that computing a matrix-vector multiplication $\mathbf{v} \mapsto \mathbf{M}\mathbf{v}$ takes almost linear time given the trapdoor. Looking at Fig. 1, it is not clear how this can help given that uncomputing the mask requires computing the inner product between the query vector $\mathbf{q}$ and a uniformly random mask vector $\mathbf{r}$. We recall, however, that when we fold the protocol, say twice, the query vector is not just any arbitrary vector, but rather a tensor product of two vectors with square-root the length, i.e. $\mathbf{q} = \mathbf{q}_1 \otimes \mathbf{q}_2$. Hence, by arranging the mask $\mathbf{r}$ into a square matrix $\mathbf{R}$, the product $\mathbf{q}^{\intercal}\mathbf{r}$ may be written as $\mathbf{q}_1^{\intercal}\mathbf{R}\mathbf{q}_2 = \mathbf{q}_1^{\intercal}(\mathbf{R}\mathbf{q}_2)$. Consequently, the client may generate $\mathbf{R}$ as a trapdoored matrix, which lets him compute $\mathbf{R}\mathbf{q}_2$ in time almost linear in the length of $\mathbf{q}_2$ and, therefore, the product $\mathbf{q}^{\intercal}\mathbf{r}$ in time square-root that of the naive implementation.

We do not know how to use trapdoored matrices to obtain greater saving in client runtime when we fold the protocol further: when $\mathbf{q}$ is, for instance, a 3-fold tensor, then a trapdoored matrix gives a more efficient inner product along one dimension out of three, making the relative runtime saving worse ($\approx n^{2/3}$ when the size of an encoded database row is n). In any case, when applying the above optimization to a balanced 2-folded protocol, where the database has $N^{1/3}$ rows, each encoded by a 2-fold tensor code of dimension $N^{1/3} \times N^{1/3}$, we obtain a protocol where uncomputing the mask takes time only $O(N^{1/3+\varepsilon})$. (Recall such a protocol has $O(N^{1/3+\varepsilon})$ communication.)

It remains to show how to make query generation efficient as well: recall query generation in a 2-folded protocol naively takes time $O(M^{2/t}) = O(M)$, where M is the length of an encoded database row and is $O(N^{2/3+\varepsilon})$ in our case. Since generating a codeword from the dual code corresponds to multiplying the generator matrix of the code with a random vector, we could potentially use trapdoored matrices here as well. There is, however, a more direct way: instead of using a uniformly random code, we use a random code where encoding can be done in almost linear time, without compromising the security of the protocol.

We exploit the fact that LPN remains as hard when replacing the random LPN secret with a secret that is sampled so that every coordinate is Bernoulli [ACPS09]. We observe that when basing sparse-noise LSN on this modified variant of LPN (see Theorem 1.2), we obtain a sparse-noise LSN where the hidden code is not a uniformly random code, but rather a random code with a sparse generator matrix, i.e. a matrix with low Hamming weight. We can set the parameters such that the generator matrix is sufficiently sparse to give encoding time almost linear, concretely $O(N^{1/3+\varepsilon})$ in the above instance of the protocol. Overall, we obtain an LPN-based protocol where both communication and client runtime are $O(N^{1/3+\varepsilon})$.

3 Preliminaries

Linear algebra. Throughout this paper, we use $\mathbb{F}$ to denote a general finite field, but refer to $\mathbb{F} = \mathbb{F}_2$ by default. While our definitions, conjectures and constructions are meaningful over general fields, we will only need to apply them over $\mathbb{F}_2$. We denote by $\mathtt{Rank}_r(\mathbb{F}^{n\times m})$ the set of $n \times m$ matrices over $\mathbb{F}$ of rank r and by $\mathbf{Span}(M)$ the row-span of a matrix M over the underlying field $\mathbb{F}$.

Representing linear and affine spaces. For a linear k-dimensional subspace $V \subset \mathbb{F}^n$, we will denote by $\langle V \rangle$ a random matrix $\mathbf{V} \in \mathbb{F}^{n\times k}$ that spans V by its rows, i.e. $\mathbf{Span}(\mathbf{V}) = V$. Note that $\langle V \rangle$ can be efficiently sampled given any basis for V. For a product space $V = V_1 \times \ldots \times V_s$ we let $\langle V \rangle = (\langle V_1 \rangle, \ldots, \langle V_s \rangle)$. Finally, for an affine subspace $A = \mathbf{v} + V$, we will denote by $\langle A \rangle$ the randomized representation $(\mathbf{a}, \langle V \rangle) \in \mathbb{F}^{n\times(k+1)}$ for a uniformly random $\mathbf{a} \leftarrow A$. Note that $\langle \mathbf{v} + V \rangle$ hides the representative $\mathbf{v}$.

Probability. We let U_n denote the uniform distribution over $\{0,1\}^n$ and use $x \leftarrow X$ to denote a uniformly random choice of x from a set X. We write $X \equiv Y$ to indicate that X and Y are identically distributed. We denote by $\mathsf{Ber}(\mu)$ a Bernoulli random variable which takes the value 1 with probability μ and 0 with probability $1-\mu$. More generally, for a finite field $\mathbb{F}$, the Bernoulli variable $\mathsf{Ber}_{\mathbb{F}}(\mu)$ takes the value 0 with probability $1-\mu$ and is otherwise uniformly random in $\mathbb{F} \setminus \{0\}$. The subscript $\mathbb{F}$ will be omitted when it is clear from the context. A probability sequence $\delta := \delta(\lambda)$ is *negligible* if $\delta(\lambda) = \lambda^{-\omega(1)}$ and *noticeable* if it is not negligible.

3.1 Standard Cryptographic Notions

We use λ to denote a computational security parameter and require by default security against non-uniform poly(λ)-time adversaries.

Definition 3.1 (Computational Indistinguishability)**.** *We say that a pair of distribution ensembles* $\{X_\lambda\}_{\lambda\in\mathbb{N}}$ *and* $\{Y_\lambda\}_{\lambda\in\mathbb{N}}$ *are* computationally indistinguishable*, and write* $X \approx_c Y$*, if for every non-*

uniform $\mathrm{poly}(\lambda)$*-time distinguisher* $\mathcal{A}$ *there is a negligible* δ *such that for every* $\lambda \in \mathbb{N}$

$$\Pr[D(X_\lambda) = 1) - \Pr[D(Y_\lambda) = 1] \leq \delta(\lambda).$$

We say that a distribution is *pseudorandom* if it is computationally indistinguishable from the uniform distribution over its domain.

Definition 3.2 (Pseudorandom Function)**.** *A function* $F(\mathsf{sk}, x)$ *is called a* pseudorandom function (PRF) *if it satisfies the following.*

1. ***Syntax:*** *F is polynomial-time computable; moreover for any $\lambda \in \mathbb{N}$, key $\mathsf{sk} \in \{0,1\}^\lambda$ and input $x \in \{0,1\}^\lambda$, we have $F(\mathsf{sk}, x) \in \{0,1\}^\lambda$.*

2. ***Pseudorandomness:*** *For any non-uniform polynomial-time algorithm $\mathcal{A}$ that has access to an oracle, the advantage*

$$\mathsf{Adv}(\lambda) = \left|\Pr[\mathcal{A}_\lambda^{F(\mathsf{sk},\cdot)} = 1] - \Pr[\mathcal{A}_\lambda^{R(\cdot)} = 1]\right|$$

is negligible, where sk is uniform in $\{0,1\}^\lambda$, $R : \{0,1\}^\lambda \to \{0,1\}^\lambda$ is a truly random function, and $\mathcal{A}_\lambda$ can only query its oracle on inputs of length λ.

A PRF as above is implied by any one-way function, and implies a PRF with arbitrary polynomial input and output lengths [Gol01]. One-way functions are implied by all of the assumptions we consider in this paper.

3.2 Secret-key Private Information Retrieval

The following definition of secret-key PIR (sk-PIR) is adapted from [BIPW17, CHR17]. Informally, an sk-PIR protocol is a variant of standard (single-server) PIR in which the client has a secret key which is used to encode the database. More concretely, in an offline preprocessing phase, the client randomly encodes the database $x \in \{0,1\}^N$ into an encoded database $\widehat{x} \in \{0,1\}^{\widehat{N}}$ using a secret key sk and stores $\widehat{x}$ on a remote server. In each invocation of the online phase, the client can use sk to retrieve a bit x_i by exchanging $o(N)$ bits with the server, without revealing i to the server.

Definition 3.3 (Secret-key PIR)**.** *A* secret-key private information retrieval *protocol (or* sk-PIR *for short) is a tuple of PPT algorithms* $\mathsf{skPIR} = (\mathsf{G}, \mathsf{E}, \mathsf{Q}, \mathsf{A}, \mathsf{D})$ *with the following syntax:*

- $\mathsf{G}(1^\lambda, 1^N)$*: The* key generation algorithm *takes a security parameter λ and database size N and outputs a* secret key sk*. We assume, w.l.o.g., that sk contains λ and N.*

- $\mathsf{E}(\mathsf{sk}, x)$*: The* database encoding algorithm *takes a secret key sk and a database $x \in \{0,1\}^N$ and outputs a database encoding $\widehat{x} \in \{0,1\}^{\widehat{N}}$.*

- $\mathsf{Q}(\mathsf{sk}, i)$*: The* query algorithm *takes a secret key sk and an index $i \in [N]$, and outputs a PIR-query $\mathsf{que} \in \{0,1\}^q$ and auxiliary information $\mathsf{aux} \in \{0,1\}^*$.*

- $\mathsf{A}(\widehat{x}, \mathsf{que})$*: The* answer algorithm *takes a database encoding $\widehat{x} \in \{0,1\}^{\widehat{N}}$ and a PIR-query que, and returns a PIR-answer $\mathsf{ans} \in \{0,1\}^a$.*

- $\mathsf{D}(\mathsf{sk}, i, \mathsf{ans}, \mathsf{aux})$*: The* decoding algorithm *takes a secret key sk, an $i \in [N]$, a PIR-answer $\mathsf{ans} \in \{0,1\}^a$ and auxiliary information $\mathsf{aux} \in \{0,1\}^*$, and outputs a bit $y \in \{0,1\}$.*

We may assume that A *and* D *are deterministic. We refer to* $\widehat{N}$ *as the* database encoding size *or* server storage*, to* q *as the* query length *and to* a *as the* answer length*.*

Correctness. *For any* $\lambda, N \in \mathbb{N}$*,* $\mathsf{sk} \in \mathsf{G}(1^\lambda, 1^N)$*,* $x \in \{0,1\}^N$*,* $\widehat{x} \in \mathsf{E}(\mathsf{sk}, x)$*,* $i \in [N]$*, and* $(\mathsf{que}, \mathsf{aux}) \leftarrow \mathsf{Q}(\mathsf{sk}, i)$*, we have* $\mathsf{D}(\mathsf{sk}, i, \mathsf{A}(\widehat{x}, \mathsf{que}), \mathsf{aux}) = x_i$*.*

Security. *Let* $m := m(\lambda)$ *be a query bound. We say that* skPIR *is* m-query secure *if for any non-uniform polynomial-time algorithm* $\mathcal{A}$ *that makes at most* $m(\lambda)$ *queries to its oracle and any polynomial* $N = N(\lambda)$*, there exists a negligible function* δ *such that for any* $\lambda \in \mathbb{N}$ *and* $x \in \{0,1\}^N$*, it holds that*

$$\left| \Pr[\mathcal{A}^{\mathsf{Q}(\mathsf{sk},\cdot)}(1^\lambda, \widehat{x}){=}1] - \Pr[\mathcal{A}^{\mathsf{Q}'(\mathsf{sk},\cdot)}(1^\lambda, \widehat{x}){=}1] \right| \leq \delta(\lambda),$$

where $\mathsf{sk} \leftarrow \mathsf{G}(1^\lambda, 1^N)$*,* $\widehat{x} \leftarrow \mathsf{E}(\mathsf{sk}, x)$ *and* $\mathsf{Q}'(\mathsf{sk}, \cdot)$ *is an oracle that ignores its input and outputs* $\mathsf{Q}(\mathsf{sk}, 1)$ *(with fresh randomness). We say that* skPIR *is* secure *if it is* m*-query secure for any polynomial query bound* $m(\lambda)$*.*

Note that the above definition only considers a single encoded database. However, it generically implies (via a standard PRF domain separation technique) a stronger notion where the same secret key can support an arbitrary number of encoded databases. Furthermore, while the definition does not require the database x to be hidden from the server, this requirement can be enforced generically via the use of symmetric encryption; our solutions already satisfy this stronger security requirement with no additional overhead.

Complexity metrics. We will be interested in different complexity metrics of sk-PIR, including server storage $|\widehat{x}|$, communication complexity, and computational complexity of the server and the client. A major competitive advantage of our protocols compared to all prior single-server PIR protocols is minimizing the *circuit size* of the server. For the latter we consider the standard measure of counting gates in a circuit implementing $\mathsf{A}(\widehat{x}, \mathsf{que})$ over the B_2 basis, namely where a gate can compute any function $g : \{0,1\}^2 \to \{0,1\}$. Finally, similarly to prior works on PIR with preprocessing, we will also consider the *cell probe* complexity of the server, namely the number of bits from $\widehat{x}$ that $\mathsf{A}(\widehat{x}, \mathsf{que})$ needs to read. Some of our construcctions will achieve slightly sublinear cell-probe complexity, thus realizing a weak notion of "doubly efficient PIR" with secret-key preprocessing.

3.3 Learning Parity with Noise

We define the *learning parity with noise* problem (LPN) [BFKL94] over a general field $\mathbb{F}$. LPN is parameterized by a *dimension* parameter $k := k(\lambda)$, a *noise rate* $\varepsilon := \varepsilon(\lambda) \in (0,1)$ and a *number of samples* given to the adversary which we denote by $m := m(\lambda)$. When $\mathbb{F}$ is not specified, it is assumed that $\mathbb{F} = \mathbb{F}_2$ by default, in which case $\mathsf{Ber}_{\mathbb{F}}(\varepsilon)$ is a standard Bernoulli variable.

Definition 3.4 (LPN)**.** *The* (decisional) learning parity with noise *assumption* (k, ε)-LPN *over* $\mathbb{F}$ *asserts that for any polynomial* $m := m(\lambda)$ *we have:*

$$(\mathbf{a}_i, \mathbf{a}_i^\mathsf{T}\mathbf{s} + r_i)_{i\in[m]} \quad \approx_c \quad (\mathbf{a}_i, u_i)_{i\in[m]}\,,$$

where $\mathbf{s} \leftarrow \mathbb{F}^k$ *and, for any* i*,* $\mathbf{a}_i \leftarrow \mathbb{F}^k$*,* $r_i \leftarrow \mathsf{Ber}_{\mathbb{F}}(\varepsilon)$*, and* $u_i \leftarrow \mathbb{F}$*. We consider three noise regimes:*

- *In* high-noise LPN *we assume that* (k, ε)-LPN *holds with* $\varepsilon = 1/k^\gamma$ *for some* $0 < \gamma < 1/2$*.*
- *In* mid-noise LPN *we assume that* (k, ε)-LPN *holds with* $\varepsilon = 1/k^\gamma$ *for every* $\gamma < 1$*.*

- *In* low-noise LPN *we assume that* (k, ε)-LPN *holds with* $\varepsilon = \log^c(k)/k$ *for some* $c > 1$.

We denote by (k, ε, m)-LPN *a restricted variant of LPN where indistinguishability holds only with* $m := m(\lambda)$ *samples.*

While mid-noise (and hence low-noise) LPN implies public-key encryption [Ale03], showing such an implication for high-noise LPN is a major open problem. Low-noise LPN over $\mathbb{F}_2$ with noise $\varepsilon = \log^2 k/k$ implies collision-resistant hash functions [BLVW19, YZW$^+$19]. This is open for mid-noise (or high-noise) LPN. LPN with an even lower noise of $\log^{1+\beta} k/k$, where $0 < \beta < 1$, is known to imply (plain-model) PIR with slightly sublinear communication of $N/2^{\Theta(\log^{1-\beta} N)}$ [AMR25]. Obtaining fully sublinear PIR from any flavor of LPN is open.

4 Learning Subspace with Noise

In this section we discuss different flavors of the *learning subspace with noise* (LSN) assumption we introduce and use in this work.

4.1 The Basic LSN Assumption

We start by discussing the basic LSN variant from Definition 1.1. Recall that for a dimension parameter $k := k(\lambda)$ and noise rate parameter $\mu = \mu(\lambda)$, the (k, n, μ)-LSN problem is to distinguish between the following two experiments. In the pseudorandom experiment, we are given a polynomial $m := m(\lambda)$ number of samples from a secret k-dimensional random linear code $C \subseteq \mathbb{F}^n$, where each sample $\mathbf{c}_i$ is replaced by a uniformly random vector $\mathbf{u}_i \in \mathbb{F}^n$ with probability μ. In the random experiment, we are just given m random vectors $\mathbf{u}_i \in \mathbb{F}^n$.

The search version of the above problem, namely recovering C from the noisy samples, can be viewed as an instance of the extensively studied problem of *learning mixtures* of simple distributions [Vid03, FSO06, RSS14, CM19]. A mixture of two distributions D_0 and D_1 samples an element from D_0 with certain probability μ and an element from D_1 with probability $1 - \mu$. The general task of learning mixtures is to learn information about the distributions D_0 and D_1 given samples from their mixture. LSN specifically considers the decision problem of distinguishing a mixture of D_0 and D_1 from the uniform distribution over their domain, where D_0 is the uniform distribution over $\mathbb{F}^n$ and D_1 is the uniform distribution over a random dimension-k subspace C. More concretely, LSN is a decision variant of the problem of *learning mixtures of subspaces over a finite field*, which was previously studied by Chen, De, and Vijayaraghavan [CDV21].

The main relevant results from [CDV21] are that: (1) the search version of LSN (hence also the decision version) can be solved in time $n^{O(\log(1/(1-\mu))/(1-\rho))}$ where $\rho = k/n$, and (2) over $\mathbb{F}_2$, the search version of the $(k, k+1, \mu)$-LSN assumption is equivalent to the $(k, \mu/2)$-LPN assumption.

In the following sections we present the following results about (decision) LSN:

- In Section 4.1.1 we extend the parameter regime in which LSN implies LPN, showing how to break LSN with a minimal number of samples using an LPN oracle.
- In Section 4.1.2 we present the "tensored rank" distinguisher for LSN, which is a simplified version of the algorithm for the search problem from [CDV21].

4.1.1 LSN implies LPN

In the following we complement the "LSN implies LPN" result from [CDV21, Proposition 21], which holds for $n = k+1$, by a similar implication that holds for a more general parameter regime.

Here we consider $\mathbb{F} = \mathbb{F}_2$ by default, though the proof extends to any $\mathbb{F}$ such that $|\mathbb{F}| \leq \mathrm{poly}(\lambda)$ (for which LPN is known to have a search-to-decision reduction).

Proposition 4.1 (LSN implies LPN). *The (k, n, μ)-LSN assumption over $\mathbb{F}$ with $n > k + 3\log(k/(1-\mu))$ and $\mu > 0.78$ implies the (k', ε, m)-LPN assumption over $\mathbb{F}$ with $k'(\lambda) = k/(1-\mu) - n$, $\varepsilon = (1 - 1/|\mathbb{F}|)(1-\mu)$ and $m(\lambda) = k/(1-\mu) = k' + n$.*

We prove the above implication via a connection between the hardness of LPN and the hardness of syndrome decoding for linear codes (aka *dual-LPN*) (see, e.g., [BCG+19, Section 3.3]). We need a strong version of the statement "hardness of syndrome decoding implies LPN," which we derive by amplifying a search-to-decision reduction from [AIK07].

Lemma 4.1 (From Syndrome Decoding to LPN). *Let $k := k(\lambda) > \lambda$ and $m := m(\lambda)$ be polynomials such that $m > (1+\delta)k$ for some constant $\delta > 0$, and let $\varepsilon(\lambda) > 0$. Assume that for every non-uniform $\mathrm{poly}(\lambda)$-time algorithm $\mathcal{A}$, there is a noticeable δ such that for every $\lambda \in \mathbb{N}$*

$$\Pr[\mathcal{A}(\mathbf{H}, \mathbf{y}) = \mathbf{e}] \leq 1 - \delta(\lambda),$$

where $\mathbf{H} \leftarrow \mathbb{F}^{(m-k)\times m}$ and $\mathbf{y} = \mathbf{He}$ for $\mathbf{e} \in \mathbb{F}^m$ sampled from $\mathsf{Ber}^m(\varepsilon)$ conditioned on its Hamming weight is $\varepsilon \cdot m$. Then, the (k, ε, m)-LPN assumption holds.

We defer the proof of Lemma 4.1 to Appendix B and proceed to prove Proposition 4.1. The high level idea is that if LPN fails, then we have an efficient syndrome decoder that succeeds with overwhelming probability to recover a random sparse error vector from its syndrome. We can use this decoder in the LSN experiment to recover the first (sparse) linear dependence between samples from the hidden code C with inverse-polynomial success probability, which suffices for breaking (decision) LSN with inverse-polynomial advantage. We proceed with the formal details.

Proof. It suffices to show that LSN implies the condition of Lemma 4.1. Assume, towards contradiction, the existence of an algorithm $\mathcal{A}$ that finds $\mathbf{e}$ given $(\mathbf{H}, \mathbf{He}) \in \mathbb{F}^{(m-k')\times m} \times \mathbb{F}^{m-k'}$ with probability $1 - \delta$ for a negligible δ. We build a distinguisher against (k, n, μ)-LSN given $3(m+1)+1$ samples over $\mathbb{F}^n$, which we denote by $\mathbf{v}_1^{(j)}, \ldots, \mathbf{v}_{m+1}^{(j)}$ for $j \in \{1,2,3\}$ and $\mathbf{v}^*$. The distinguisher performs the following experiment three times, with $(\mathbf{v}_1, \ldots, \mathbf{v}_{m+1}) := (\mathbf{v}_1^{(j)}, \ldots, \mathbf{v}_{m+1}^{(j)})$ for $j \in \{1,2,3\}$: arrange the first m samples $\mathbf{v}_1, \ldots, \mathbf{v}_m$ in the columns of a matrix $\mathbf{H} \in \mathbb{F}^{n \times m}$ (note $m - k' = n$) and send the pair $(\mathbf{H}, \mathbf{v}_{m+1})$ to $\mathcal{A}$ to receive an output $\mathbf{e} \in \mathbb{F}^m$ with Hamming weight k.

Let $C^{(j)}$ denote the span of $\{\mathbf{v}_i \mid \mathbf{e}_i \neq 0\}$ in each of the experiments. The distinguisher returns 1 if and only if $C = \mathbf{Span}(C^{(1)} \cup C^{(2)} \cup C^{(3)})$ has rank k and $\mathbf{v}^* \in C$.

If the distinguisher is given the uniform distribution and, in particular, $\mathbf{v}^*$ is uniformly random, then the probability that it is in any fixed rank-k subspace is at most $|\mathbb{F}|^{k-n} = O\big((1-\mu)^3/k^3\big)$.

Assume the distinguisher is given LSN samples corresponding to a random hidden subspace $C \subset \mathbb{F}^n$ of dimension k. We first analyze each execution of the (sub-)experiment separately.

Recall the samples distribute by $\mathbf{v}_i \leftarrow C$ with probability $1 - \mu$ and $\mathbf{v}_i \leftarrow \mathbb{F}^n$ with probability μ. Let $S = \{i \mid \mathbf{v}_i \in C\}$. We denote by E the event where the following three conditions are met: $|S| = k$, $\mathbf{rank}(\{\mathbf{v}_i \mid i \in S\}) = k$ and $\mathbf{v}_{m+1} \in C$. Notice that the size of S is a variable that follows the Binomial distribution with expectation $(1-\mu)m = k$. It holds that $\Pr[|S| = k] = \binom{m}{k} 2^{-kH(1-\mu)} \geq 1/k$. The probability that k uniform samples from a k-dimensional subspace C are linearly independent is $\prod_{i=0}^{k-1}(1 - |\mathbb{F}|^{i-k}) \geq 0.28$. Lastly, $\mathbf{v}'_{m+1} \in C$ with probability at least $1 - \mu$ by definition. Overall, E occurs with probability at least $\Omega((1-\mu)/k)$. Conditioned on E, we may rearrange the sampling of $(C, \mathbf{v}_1, \ldots, \mathbf{v}_{m+1})$ as follows:

1. Sample $b \leftarrow \mathsf{Ber}^m(1-\mu)$ conditioned on $|b| = k$ and let $S = \{i \mid b_i = 1\}$.
2. Start with $C = \{\mathbf{0}\}$.
3. For $i = 1, \ldots, m$:
   - If $i \notin S$, sample $\mathbf{v}_i \leftarrow \mathbb{F}^n$.
   - If $i \in S$, sample $\mathbf{v}_i \leftarrow \mathbb{F}^n \setminus C$ and add it to C, i.e. $C = \mathbf{Span}(C \cup \mathbf{v}_i)$.
4. Sample $\mathbf{v}_{m+1} \leftarrow C$.

We may further rewrite the sampling of $\mathbf{v}_{m+1}$ as:

4. Sample $\mathbf{v}_{m+1} = \sum_{i \in S} \beta_i \cdot \mathbf{v}_i$ for $\beta_i \leftarrow \mathbb{F}$.

Since the probability that a uniformly random vector over $\mathbb{F}^n$ is in a k-dimensional subspace is $|\mathbb{F}|^{k-n} = 2^{-\Omega(k)}$, the above experiment is statistically close to an experiment where the vectors $\mathbf{v}_1, \ldots, \mathbf{v}_m$ are sampled uniformly at random. Let E' denote the condition that there are exactly $1 - 1/|\mathbb{F}|$ non-zero β_i's. Once again, we have $\Pr[E'] = \Omega(1/m) = \Omega((1-\mu)/k)$ and, therefore, $\Pr[E', E] = \Omega((1-\mu)^2/k^2)$. Conditioned on E and E', $(\mathbf{v}_1, \ldots, \mathbf{v}_{m+1})$ is statistically close to the distribution $(\mathbf{H}, \mathbf{y})$ from the lemma with $\varepsilon = (1-\mu)(1-1/|\mathbb{F}|)$, and, upon feeding them as input to $\mathcal{A}$, we obtain $\mathbf{e}$ such that $\sum_i \mathbf{e}_i \mathbf{v}_i = \mathbf{v}_{m+1}$ by assumption. Further, $\mathbf{e}$ corresponds to $\mathbf{e}_i = \beta_i \cdot b_i$ since, by a simple union bound, this is the only solution with probability all but $\binom{m}{k}|\mathbb{F}|^{k-m} \leq 2^{m(H(\mu)-\mu)}$, which is negligible in m when $\mu > 0.78$ (where $H(\mu) - \mu = \Omega(1)$). Therefore, $\mathbf{e}$ reveals a random subspace of C of dimension $(1-1/|\mathbb{F}|)k$ conditioned on E and E'. Since this occurs with probability all but negligible in the experiment conditioned on these events, it does also in the original experiment (since E and E' both have inverse-polynomial probability). Hence, when E and E' occur, the distinguisher obtains three random subspaces of C of rank $(1-1/|\mathbb{F}|)k$ each, which cover C entirely with probability all but negligible (this is the probability that $3(1-1/|\mathbb{F}|)k \geq 1.5k$ vectors from C have rank k).

Overall, we obtain a distinguishing advantage of $\Omega((1-\mu)^2/k^2 - (1-\mu)^3/k^3)$, which is inverse-polynomial. $\square$

4.1.2 A Tensored-Rank Distinguisher for LSN

The previous "LSN implies LPN" claim should be interpreted as saying that LSN is at least as strong of an assumption as LPN. However, it does not show that LSN is strictly stronger. In particular, it does not rule out the possibility that (k, n, μ)-LSN holds with constant code rate $0 < k/n < 1$ and noise rate $0 < \mu < 1$, for which the corresponding LPN parameters are considered to be conservative.

In the following we show a simplified version of the learning algorithm from [CDV21] that leverages a larger number of samples than the LPN-based attack to break the (decision) LSN assumption with any positive constant noise rate μ.[4] In light of this algorithm, we must require $1 - \mu$ to be sub-constant in Conjecture 1.1.

Proposition 4.2 (Rank attack against LSN with low noise rate). *For any constant $d \in \mathbb{N}$, field $\mathbb{F}$, polynomials k, n and noise rate $\mu < 1 - (k/n)^d$, there is a polynomial-time distinguisher that breaks (k, n, μ)-LSN in $n^{O(d)}$ time using $m = O(n^d)$ samples.*

[4]Notably, our algorithm has smaller sample complexity compared to the learning algorithm from [CDV21]. While their algorithm requires $O(n^d)$ samples for $d = \Omega(\log(1/(1-\mu))/1-\rho)$, ours needs only $d = O(\log(1/(1-\mu))/\log(1/\rho)$.

Proof. Let $q = |\mathbb{F}|$. The distinguisher is given $m = 1 + (4qn)^d \log q$ samples $\mathbf{v}_1, \ldots, \mathbf{v}_m \in \mathbb{F}^n$ which are either uniformly random or come from a random rank-k subspace generated by $\mathbf{C} \leftarrow \mathbb{F}^{k \times n}$, mixed with noise. The distinguisher computes the d-fold tensor product of each sample $\widehat{\mathbf{v}}_i = \mathbf{v}_i \otimes \cdots \otimes \mathbf{v}_i$ and arranges all of the obtained vectors in the rows of a $m \times n^d$ matrix $\mathbf{V}$. The distinguisher then outputs 1 if and only if $\mathbf{rank}(\mathbf{V}) = n^d$.

If the $\mathbf{v}_i$ are LSN samples of the form $\mathbf{v}_i = \mathbf{a}_i^\intercal \mathbf{C} + \mathbf{e}_i$ for random $\mathbf{a}_i \leftarrow \mathbb{F}^k$ and $\mathbf{e}_i$ that is nonzero with probability μ, then, in expectation, there are $(1-\mu)m > 2k^d$ noiseless samples where $\mathbf{v}_i = \mathbf{a}_i^\intercal \mathbf{C}$. Since the noise in the samples is i.i.d., we may derive by Chernoff that there are at least $\frac{3}{2}k^d$ noiseless samples with probability at least $e^{-\Omega(k^d)}$. Assuming this is the case, there exist at least $\frac{3}{2}k^d$ rows in $\mathbf{V}$ that come from the subspace spanned by the d-fold tensor product $\mathbf{C} \otimes \cdots \otimes \mathbf{C}$, which has rank only k^d. Hence, the rank of $\mathbf{V}$ cannot be full.

On the other hand, we argue that if the samples $\mathbf{v}_i$ are uniform, then the rank of $\mathbf{V}$ is full with probability at least $1 - 1/e$. To see this, notice that the existence of a linear combination $\mathbf{w} \in \mathbb{F}^{n^d}$ satisfying $\mathbf{V}\mathbf{w} = \mathbf{0}$ implies the existence of a degree-d polynomial over n variables that vanishes on all $\mathbf{v}_1, \ldots, \mathbf{v}_m$. Indeed, letting $P_{\mathbf{w}}$ denote the polynomial with coefficients defined by $\mathbf{w}$, it holds that $(\mathbf{v}_i \otimes \cdots \otimes \mathbf{v}_i)^\intercal \mathbf{w} = P_{\mathbf{w}}(\mathbf{v}_i)$. It is sufficient to bound, then, the probability that m uniformly random assignments to n variables over $\mathbb{F}$ are all roots of some degree-d polynomial.

A known fact (see, e.g. [KLP68]) is that an n-variate degree-d polynomial over $\mathbb{F}$ has at most $q^n - q^{n-d}$ roots. (The simple case where $\mathbb{F} = \mathrm{GF}(2)$ is implied by the distance analysis of binary Reed-Muller codes). Thus, the probability that $(4qn)^d$ uniform assignments are all roots of a given polynomial is at most $(1 - 1/q^d)^m = e^{-m/q^d} = e^{-1-(4n)^d \log q}$. The number of all such polynomials is bounded by $q^{\binom{n}{n+d}} \le q^{((en/d)+1)^d} < e^{(4n)^d \log q}$. Hence, the probability that there exists a polynomial that vanishes on all $\mathbf{v}_i$ is less than $1/e$ by union bound. □

4.2 Split-LSN

Our most efficient sk-PIR constructions rely on the split-LSN conjecture (Conjecture 1.2), which we put forward in this work. One source of belief in this conjecture comes from the analogy with the assumption used in previous works on sk-PIR [BIPW17, CHR17] (see Section 1.6), which seems to be more structured. Here we provide an additional heuristic justification by considering a natural class of algebraic distinguishers that capture the previous attack on basic LSN.

The tensored rank attack of Proposition 4.2 can be seen as finding a code-dependent nonzero degree-d polynomial that vanishes (with high probability) on all samples from the pseudorandom experiment but not on samples from the random experiment. This abstraction also captures the algorithm of Arora and Ge [AG11] for LPN with structured noise. To consider this attack in the context of split-LSN, define for each split-LSN sample $\mathbf{v} \in \mathbb{F}^\ell$ the generalized d-tensor $\mathbf{v}^{\le d}$ that includes the values of all products of at most d entries of $\mathbf{v}$. (For the parameters of our split-LSN conjecture, we have $\ell = 2n$.) Now, suppose that for the secret code $C \subset \mathbb{F}^n$ there is a nonzero polynomial p_C of (total) degree $\le d$ that vanishes on all split-LSN samples $\mathbf{v}_i \in \mathbb{F}^\ell$ in the pseudorandom experiment. Such a polynomial will distinguish these samples from random samples in $\mathbb{F}^\ell$. Moreover, the existence of such a polynomial implies that all tensored samples $\mathbf{v}_i^{\le d}$ in the pseudorandom experiment satisfy the same linear dependency, which in turn implies a rank deficiency of the d-tesnored sample matrix. In the converse direction, a rank deficiency in the d-tensored samples implies the existence of such a code-dependent nonzero polynomial p_C of degree$\le d$ that vanishes on the pseudorandom samples.

We do not have a rigorous analysis of the minimal degree distribution of p_C corresponding to our split-LSN conjecture. However, we have run preliminary experiments suggesting that indeed this degree grows even when just moderately increasing the splitting parameter s beyond the inverse-rate n/k. We leave a more thorough analysis of our LSN-related conjectures to future work.

4.3 A Unified LSN Framework

Our general template for constructing secret-key PIR protocols can be instantiated using different flavors of the LSN assumption that yield different efficiency features. To present and analyze these different variants in a unified way, we formulate a generalized variant of LSN that captures all of the LSN flavors we will use as special cases.

More concretely, the useful LSN flavors include standard LSN (Definition 1.1) split-LSN (Definition 1.2) and a new notion of sparse-noise LSN (see Definition 4.3 below) that will serve as a convenient step towards our LPN-based results.

All these LSN flavors assert the indistinguishability between random codewords $\mathbf{c}_i$ sampled from a secret k-dimensional code $C \subset \mathbb{F}^n$ and uniformly random samples from $\mathbb{F}^n$, where the samples are hidden using some kind of noise. Syntactically, in the above assumptions the noisy version of $\mathbf{c}_i$ is either a vector in $\mathbb{F}^n$ (for standard and sparse-linear LSN) or alternatively it is an *affine subspace* of $\mathbb{F}^n$ (for regular LSN and split-LSN). To capture both cases by the same definition, we consider the noise to always be an affine subspace $E_i \subset \mathbb{F}^n$ which is added to $\mathbf{c}_i$ in the usual sense. Namely, $\mathbf{c}_i + E_i$ is an affine space of the same dimension as E_i. The noise spaces E_i will be independently generated by a sampler $\mathcal{E}$, where $\mathcal{E}$ is generated jointly with the code C. Finally, we will also allow sampling the code C from a general distribution over k-dimensional linear codes; jumping ahead, this will be useful for reducing the client's runtime. We further allow the random code and the noise sampler $\mathcal{E}$ to be correlated, and therefore consider a joint distribution $\mathcal{D}$ over C and $\mathcal{E}$.

In the following definition, we denote by $k := k(\lambda)$ and $n := n(\lambda)$ the dimension and length of the secret code, both polynomial. We will let $\mathcal{D} := \mathcal{D}(1^\lambda, 1^k, 1^n)$ denote a sampling algorithm that jointly samples a code C and and noise distribution $\mathcal{E}$, whose output is an affine space.

Definition 4.1 (General LSN). *An* LSN sampler *is a PPT algorithm $\mathcal{D}(1^\lambda, 1^k, 1^n)$ that outputs a pair $(C, \mathcal{E})$ with the following syntax: $C \subset \mathbb{F}^n$ is a k-dimensional linear code and $\mathcal{E}$ is a probabilistic circuit sampling a description $\langle E\rangle$ of an affine subspace $E \subset \mathbb{F}^n$. For dimension parameters $k := k(\lambda)$ and $n := n(\lambda)$, the $(k, n, \mathcal{D})$-LSN assumption asserts that for secret $(C, \mathcal{E}) \leftarrow \mathcal{D}(1^\lambda, 1^{k(\lambda)}, 1^{n(\lambda)})$ and for any polynomial number of samples $m := m(\lambda)$ we have:*

$$(\langle \mathbf{c}_i + E_i\rangle)_{i\in[m]} \approx_c (\langle \mathbf{u}_i + E_i\rangle)_{i\in[m]},$$

where for each i we take independent samples $\mathbf{c}_i \leftarrow C$, $\mathbf{u}_i \leftarrow \mathbb{F}^n$ and $E_i \leftarrow \mathcal{E}$.

We now derive the LSN variants from Definitions 1.1 and 1.2 as well as a new sparse-noise LSN variant as instances of general LSN.

Definition 4.2 (LSN Variants). *We define the following $(k, n, \mathcal{D})$-LSN instances, where $\mathcal{D}$ outputs a uniformly random k-dimensional code $C \subset \mathbb{F}^n$ and $\mathcal{E}$ as defined below.*

- *In the standard LSN assumption (k, n, μ)-LSN, $\mathcal{E}$ is sampled independently of C as follows:*
  - *with probability μ, $\mathcal{E}$ outputs $E = \{\mathbf{r}\}$ for a uniformly random $\mathbf{r} \leftarrow \mathbb{F}^n$,*
  - *with probability $1 - \mu$, $\mathcal{E}$ deterministically outputs $E = \{0^n\}$.*
- *In the sparse-noise LSN assumption (k, n, ε)-snLSN, $\mathcal{E}$ is defined by a uniformly random set of coordinates $I \subset [n]$ of size k subject to the constraint $C_I = \mathbb{F}^k$. $\mathcal{E}$ outputs $E = \{\mathbf{r}\}$ where $\mathbf{r}_i = 0$ for $i \in I$ and $\mathbf{r}_i \leftarrow \mathsf{Ber}(\varepsilon)$ otherwise.*

- *In the split-LSN assumption (k, n, r, s)-SLSN, $\mathcal{E}$ always samples uniform $(r-1)$-dimensional linear subspaces $E_1, \ldots, E_s \subset \mathbb{F}^{n/s}$ and outputs $E = E_1 \times \cdots \times E_s$.*

The security and complexity features of our sk-PIR protocols will be inherited by the features of the underlying LSN assumption.

4.4 LPN Implies Sparse-Noise LSN

Our LPN-based protocols will be derived as instances of our general LSN-based framework, using the sparse-noise LSN variant. In this section we show that this variant of LSN is implied by standard LPN with corresponding parameters.

Recall that in sparse-noise LSN, the noise is not uniformly random with probability μ and zero otherwise, as it is the case for standard LSN (Definition 1.1). Instead, we pick a random set of k linearly independent columns of a generating matrix of C and flip each *other* coordinate with probability μ. For convenience, we give a self-contained definition below.

Definition 4.3 (Sparse-noise LSN)**.** *The* sparse-noise LSN *assumption* (k, n, ε)-snLSN *states that for a uniformly random secret rank-k matrix $\mathbf{C} \in \mathbb{F}^{k\times n}$ and any polynomial number of samples $m := m(\lambda)$, it holds that:*

$$(\mathbf{c}_i + (0^k, \mathbf{r}_i)^\intercal \cdot \Pi)_{i\in[m]} \overset{c}{\approx} (\mathbf{u}_i)_{i\in[m]},$$

where $\mathbf{c}_i = \mathbf{a}_i^\intercal \mathbf{C}$ for $\mathbf{a}_i \in \mathbb{F}^k$, Π is a uniformly random permutation matrix conditioned on $\mathbf{C}\cdot\Pi^{-1} = (\mathbf{R}, \mathbf{U})$ for some $\mathbf{R} \in \mathtt{Rank}_k(\mathbb{F}^{k\times k})$, $\mathbf{r}_i \leftarrow \mathsf{Ber}^{n-k}(\varepsilon)$ and $\mathbf{u}_i \leftarrow \mathbb{F}^n$.

We remark the following regarding sparse-noise LSN:

(i) With probability $(1-\varepsilon)^{n-k}$, we have $\mathbf{r}_i = 0^{n-k}$. Thus, one may think of sparse-noise LSN as a generalization of LSN with noise rate $\mu = 1 - (1-\varepsilon)^{n-k} \approx 1 - e^{\varepsilon(k-n)}$, which is approximately $\varepsilon(n-k)$ in the vanishing regime. For instance, sparse-noise LSN with $\varepsilon = 1/k^\gamma$ and $n = k + o(k^\gamma)$ for some $0 < \gamma < 1/2$, which can be based on high-noise LPN, induces LSN noise rate $\mu = o(1)$.

(ii) An alternative way to view the joint distribution over $\mathbf{C}$ and Π, which also provides an efficient sampling procedure, is to sample a uniformly random permutation matrix Π and to define $\mathbf{C}$ as $\mathbf{R}(I, \mathbf{S}) \cdot \Pi$, where I is the $k \times k$ identity, $\mathbf{S} \leftarrow \mathbb{F}^{k\times(n-k)}$ is uniform and $\mathbf{R} \leftarrow \mathtt{Rank}_k(\mathbb{F}^{k\times k})$. In fact, from this angle, it is easy to see the reduction from LPN since $\mathbf{a}_i^\intercal\mathbf{C} + (0^k, \mathbf{r}_i)^\intercal\Pi$ distributes just like $(\mathbf{a}_i^\intercal(I, \mathbf{S}) + (0^k, \mathbf{r}_i)^\intercal) \cdot \Pi = (\mathbf{a}_i^\intercal, \mathbf{a}_i^\intercal\mathbf{S} + \mathbf{r}_i^\intercal) \cdot \Pi$ when $\mathbf{a}_i$ is uniform. We provide more details in the proof of Proposition 4.3.

(iii) Consequently to the above remark, we may consider a less conservative version of sparse-noise LSN which still reduces from LPN, where the noise has the simpler form of $(0^k, \mathbf{r}_i)$ (without permutation) yet the code $\mathbf{C}$ is a random systematic code (recall that any code is systematic up to permuting the columns).

(iv) In the case of $k = n - 1$, sparse-noise LSN is *equivalent* to LSN with uniform noise (and half the noise rate), hence our reduction (Proposition 4.3) is a generalization of the connection made in [CDV21] for the search variants of the problems. We formalize this consequence in Corollary 4.1.

We now state our reduction from LPN to sparse-noise LSN.

Proposition 4.3 (LPN implies snLSN)**.** *The* (k,ε)-LPN *assumption implies the* (k,n,ε)-snLSN *assumption for any field* $\mathbb{F}$*, polynomials* $n(\lambda) > k(\lambda) \geq \lambda$*, and noise rate* $\varepsilon(\lambda)$*.*

A special case of Proposition 4.3, where $n = k+1$, implies a decision version of the reduction shown in [CDV21, Proposition 20]. (While the formal statement in [CDV21] refers to a reduction between the search versions, their proof shows that the LPN and LSN distributions are *equivalent* in this regime, hence implying a reduction between the decision versions of the problems as well. In fact, our proof can be seen as a generalization of their proof.)

Corollary 4.1. *For* $\mathbb{F} = \mathbb{F}_2$*, the* (k,ε)-LPN *assumption implies the* $(k,k+1,2\varepsilon)$-LSN *assumption for any polynomial* $k(\lambda)$ *and noise rate* $\varepsilon(\lambda)$*.*

Proof. The corollary follows from the observation that $(k,k+1,\varepsilon)$-snLSN is equivalent to $(k,k+1,2\varepsilon)$-LSN. To see this, notice that a sparse-noise LSN sample $\mathbf{w}_i = \mathbf{a}_i^{\intercal}\mathbf{C} + (0^k, r_i)^{\intercal} \cdot \Pi$, where $r_i \leftarrow \mathsf{Ber}(\varepsilon)$, distributes just like a (standard) LSN sample $\mathbf{v}_i = \mathbf{a}_i^{\intercal}\mathbf{C} + d_i \cdot \mathbf{r}_i^{\intercal}$ for $d_i \leftarrow \mathsf{Ber}(2\varepsilon)$ and $\mathbf{r}_i \leftarrow \mathbb{F}_2^n$: Conditioned on $d_i = 1$, which occurs with probability 2ε, $\mathbf{v}_i$ is a uniformly random vector, which is in $C = \mathbf{Span}(\mathbf{C})$ with probability half. Conditioned on $d_i = 0$, $\mathbf{v}_i$ is always in C. Overall, $\mathbf{v}_i$ is uniform in C with probability ε and is uniform in $\overline{C} = \mathbb{F}_2^n \setminus C$ with probability $1-\varepsilon$. This corresponds exactly to the distribution of $\mathbf{w}_i$: when $r_i = 0$, $\mathbf{w}_i$ is uniform in C and, otherwise, it is uniform in $C + (0^k, 1)^{\intercal}\Pi$, which is equal to $\overline{C}$ since $(0^k, 1)^{\intercal}\Pi \notin C$ because the first k columns in $\mathbf{C}\Pi^{-1}$ form a full rank matrix. □

Towards proving Proposition 4.3, we define a standard generalization of LPN (considered, e.g. in [DP12] and more explicitly in [Döt14]) where the oracle outputs $n := n(\lambda)$ samples corresponding to the same vector $\mathbf{a}_i$ and i.i.d. secret and noise vectors.

Definition 4.4 (Matrix LPN)**.** *The* (decisional) (k,n,ε)-$\mathsf{Matrix\text{-}LPN}$ assumption over $\mathbb{F}$ *states that the for any polynomial* $m := m(\lambda)$ *the following holds:*

$$(\mathbf{a}_i, \mathbf{a}_i^{\intercal}\mathbf{S} + \mathbf{r}_i)_{i\in[m]} \quad \overset{c}{\approx} \quad (\mathbf{a}_i, \mathbf{b}_i)_{i\in[m]} \, ,$$

where $\mathbf{S} \leftarrow \mathbb{F}^{k\times n}$ *and, for any* i*,* $\mathbf{a}_i \leftarrow \mathbb{F}^k$ *and* $\mathbf{b}_i \leftarrow \mathbb{F}^n$ *are uniform, and* $\mathbf{r}_i \in \mathbb{F}^n$ *is a vector of i.i.d. elements where every coordinate is* 0 *with probability* $1-\varepsilon$ *and a uniformly random field element with probability* ε*.*

The following reduction from LPN to its matrix version is a special case of [Döt14, Lemma 3.4], which is proved via a standard hybrid argument. For completeness, we include a proof in Appendix A.

Lemma 4.2 (From LPN to Matrix-LPN)**.** (k,ε)-LPN *implies* (k,n,ε)-$\mathsf{Matrix\text{-}LPN}$*, for any field* $\mathbb{F}$*, polynomials* $n(\lambda) > k(\lambda) \geq \lambda$*, and noise rate* $\varepsilon(\lambda)$*.*

We now proceed to prove the reduction from LPN to sparse-noise LSN (Proposition 4.3) using Matrix-LPN as an intermediate problem.

Proof. Let $t = n-k$. Due to Lemma 4.2, it is sufficient to show a reduction from (k,t,ε)-$\mathsf{Matrix\text{-}LPN}$. Assuming an adversary $\mathcal{A}$ that breaks (k,n,ε)-snLSN, we build an adversary $\mathcal{B}$ that breaks (k,t,ε)-$\mathsf{Matrix\text{-}LPN}$ as follows: $\mathcal{B}(1^\lambda)$ takes as input m matrix-LPN samples $(\mathbf{a}_i, \mathbf{b}_i) \in \mathbb{F}^k \times \mathbb{F}^t$. It samples a uniformly random permutation matrix $\Pi \in \mathbb{F}^{n\times n}$, and returns the output of $\mathcal{A}$ on the m samples $\mathbf{d}_i = (\mathbf{a}_i, \mathbf{b}_i)^{\intercal} \cdot \Pi$.

# in Thm. 5.1	Conjecture	Communication	Server		
			Storage	Boolean circuit size	Probes
1	High-noise LPN	$N^{\frac{2}{3}+\varepsilon}$	$N^{\frac{4}{3}+\varepsilon}$	-	-
	Mid-noise LPN	$N^{\frac{1}{2}+\varepsilon}$	$N^{1+\varepsilon}$	-	-
2	LSN (Conj. 1.1)	$N^{\frac{1}{2}+\varepsilon}$	$(1+\varepsilon)N$	-	-
		$N^{\frac{1}{2}+\varepsilon}$	$N^{1+\varepsilon}$	-	$\omega(1) \cdot N/\log N$
3	Split-LSN (Conj. 1.2)	$N^{\frac{1}{2}+\varepsilon}$	$(1+\varepsilon)N$	$(4+\varepsilon)N$	$(3/4+\varepsilon)N$

Table 1: Instantiations of our base protocol from Figure 2 under different LPN (Definition 3.4) and LSN assumptions. Here, we consider N to be also a security parameter, namely we require that any poly(N)-time adversary has advantage at most negl(N). See Theorem 5.1 for more details.

Clearly, when the samples given to $\mathcal{B}$ are uniformly random, so are the samples given to $\mathcal{A}$. Otherwise, when the samples are of the form $(\mathbf{a}_i, \mathbf{a}_i^\intercal \mathbf{S} + \mathbf{r}_i^\intercal)$, we may rewrite

$$\mathbf{d}_i = (\mathbf{a}_i, \mathbf{b}_i)^\intercal \cdot \Pi = \mathbf{a}_i^\intercal (I, \mathbf{S}) \cdot \Pi + (0^k, \mathbf{r}_i)^\intercal \cdot \Pi.$$

When $\mathbf{a}_i \leftarrow \mathbb{F}_2^k$ and $\mathbf{S} \leftarrow \mathbb{F}^{k \times t}$ are uniform, $\mathbf{a}_i^\intercal (I, \mathbf{S})$ distributes identically to $\mathbf{a}_i^\intercal \mathbf{R}(I, \mathbf{S})$ for a uniformly random full-rank matrix $\mathbf{R}$, which in turn can be written as $\mathbf{a}_i^\intercal (\mathbf{R}, \mathbf{RS})$ and is therefore equivalent to $\mathbf{a}_i^\intercal (\mathbf{R}, \mathbf{U})$ for a uniformly random $\mathbf{U} \leftarrow \mathbb{F}^{k \times t}$. Letting $\mathbf{C} = (\mathbf{R}, \mathbf{U}) \cdot \Pi$, it holds

$$\mathbf{d}_i \equiv \mathbf{a}_i^\intercal \mathbf{C} + (0^k, \mathbf{r}_i)^\intercal \cdot \Pi.$$

To complete the proof it suffices then to show that $\mathbf{C}$ and Π are distributed according to the distribution in Definition 4.3, namely a uniformly random rank-k matrix and a permutation matrix such that the first k columns in $\mathbf{C} \cdot \Pi^{-1}$ are full-rank. To that end, notice that $\mathbf{C}$ is a full-rank matrix if and only if it contains k columns that form a full-rank square matrix. Thus, one way to sample a uniformly random full-rank $k \times n$ matrix $\mathbf{C}$ is by sampling a uniformly random full-rank $k \times k$ sub-matrix $\mathbf{R}$, and then to sample k uniformly random locations for the columns of $\mathbf{R}$ in $\mathbf{C}$. Then, the rest of the columns are sampled uniformly at random. By inspection, this corresponds exactly to $\mathbf{C} = (\mathbf{R}, \mathbf{U}) \cdot \Pi$ from above. □

5 Base Protocol

We present our base sk-PIR protocol in Fig. 2. The protocol may be instantiated based on any LSN assumption satisfying the template laid in Definition 4.1.

Lemma 5.1 (Base Protocol Security)**.** *The sk-PIR protocol from Fig. 2 is secure under* $(k, n, \mathcal{D})$-LSN.

Proof. Consider a hybrid protocol where the outputs of the PRF are replaced by uniformly random outputs and, consequently, C is a uniformly random code and $\mathbf{R}$ is a uniformly random matrix. Under the security of the PRF, this is as secure as the original protocol. In the hybrid protocol, the database encoding is uniformly random in the eyes of the adversary (namely the algorithm $\mathcal{A}$ trying to break the sk-PIR, see Definition 3.3) since it is masked with a uniformly random $\mathbf{R}$. Hence, we may simulate the adversary's view only given the queries, which consist of polynomially many samples of the form $\mathbf{Q}_i = \langle \mathbf{Span}((\mathbf{v}_i + \mathbf{c}_i) + E_i) \rangle$ where $\mathbf{v}_i$ are arbitrary vectors and $\mathbf{c}_i + E_i$

Parameters:

- Functions $N_0 := N_0(N)$ and $N_1 := N_1(N)$ such that $N_0 N_1 = N$.
  // We view a database $x \in \{0,1\}^N$ as $X \in \mathbb{F}_2^{N_0 \times N_1}$ and $i \in [N]$ as $(i_0, i_1) \in [N_0] \times [N_1]$.
- Code dimension $k := k(\lambda) \geq \lambda$.
- A pseudorandom function $\mathsf{PRF} : \{0,1\}^\lambda \times \{0,1\}^\lambda \to \{0,1\}^\lambda$.

The Protocol:

- $\mathsf{G}(1^\lambda, 1^N)$: Output a uniform PRF key $\mathsf{sk} \leftarrow \{0,1\}^\lambda$.
- $\mathsf{E}(\mathsf{sk}, X)$: Use the outputs of $\mathsf{PRF}(\mathsf{sk}, \cdot)$ to determine the randomness in the following:
  1. Sample $(C, \mathcal{E}) \leftarrow \mathcal{D}(1^\lambda, 1^k, 1^n)$ for $n = k + N_1$.
     – Let $\mathbf{G} \in \mathbb{F}_2^{N_1 \times n}$ be a generator matrix of $D = C^\perp$.
     – For any $j \in [N_1]$, let $\mathbf{v}_j \in \mathbb{F}_2^n$ be a vector for which $\mathbf{G}\mathbf{v}_j \in \mathbb{F}_2^{N_1}$ is the j^{th} unit vector.
  2. Sample a uniformly random mask $\mathbf{R} \leftarrow \mathbb{F}_2^{N_0 \times n}$

  Output the following database encoding:
  $$\widehat{X} = X\mathbf{G} + \mathbf{R} \in \mathbb{F}_2^{N_0 \times n}$$
- $\mathsf{Q}(\mathsf{sk}, (i_0, i_1))$: Sample a uniform $\mathbf{c} \leftarrow C$ and $\langle E \rangle \leftarrow \mathcal{E}$. Let $\mathbf{Q} = \langle \mathbf{Span}(\mathbf{c} + \mathbf{v}_{i_1} + E) \rangle \in \mathbb{F}_2^{r \times n}$ and output
  $$\mathsf{que} = \mathbf{Q}, \;\; \mathsf{aux} = (\mathbf{Q}, \mathbf{d}),$$
  where $\mathbf{d} \in \mathbb{F}_2^r$ is the vector satisfying $\mathbf{Q}^\intercal \mathbf{d} = \mathbf{c} + \mathbf{v}_{i_1}$ if it exists or $\perp$ if it does not exist.
- $\mathsf{A}(\widehat{X}, \mathsf{que})$: Output $\mathsf{ans} = \widehat{X}\mathbf{Q}^\intercal \in \mathbb{F}_2^{N_0 \times r}$
- $\mathsf{D}(\mathsf{sk}, (i_0, i_1), \mathsf{ans}, \mathsf{aux})$: Output $y_{i_0} \in \mathbb{F}_2$ for $y = (\mathsf{ans} - \mathbf{R}\mathbf{Q}^\intercal) \cdot \mathbf{d} \in \mathbb{F}_2^{N_0}$ or $\perp$ if $\mathsf{aux} = \perp$.

Figure 2: Our Base sk-PIR Protocol under $\mathcal{D}$-LSN.

are $(k, n, \mathcal{D})$-LSN samples and, therefore, are jointly pseudorandom under the corresponding LSN assumption. Since $\mathbf{Q}_i$ is a function of $(\mathbf{v}_i + \mathbf{c}_i) + E_i$, security under LSN follows by

$$\big(\ \langle (\mathbf{v}_i + \mathbf{c}_i) + E_i \rangle\ \big)_{i \in [m]} \equiv \big(\ \langle \mathbf{v}_i + (\mathbf{c}_i + E_i) \rangle\ \big)_{i \in [m]} \overset{c}{\approx} \big(\ \langle \mathbf{v}_i + \mathbf{r}_i + E_i \rangle\ \big)_{i \in [m]} \equiv \big(\ \langle \mathbf{r}_i + E_i \rangle\ \big)_{i \in [m]} \quad (1)$$

where $\mathbf{r}_i$ is uniformly random. □

Using the split-LSN variant, where the noise rate μ is zero, the base protocol has perfect correctness and the server can be implemented by a linear-size circuit. When using the noisy variants, we obtain a noisy sk-PIR. Crucially, this is an "erasure noise," since the output is always correct unless the client outputs $\perp$, which occurs with $\leq \mu$ probability. As discussed in Section 2, the noise can be eliminated either via repetition or, for minimizing the server overhead, by applying an erasure code to each column of X.

We now formally derive our different instantiations of the base protocol.

Theorem 5.1 (Base sk-PIR from LSN Assumptions)**.** *The following holds for any constant $\varepsilon > 0$ and any $\tilde{\lambda} = \omega(\log \lambda)$:*

1. *Under $(k, 1/k^\gamma)$-LPN, for any $0 < \gamma' < \gamma$ there exists an sk-PIR protocol with storage $O(N^{2/(1+\gamma')}) + \mathrm{poly}(\lambda)$ and communication $N^{1/(1+\gamma')} + \mathrm{poly}(\lambda)$.*
2. *Under the LSN conjecture (Conj. 1.1), there exists an sk-PIR protocol with storage $(1+\varepsilon)N + \mathrm{poly}(\lambda)$ and communication $\tilde{\lambda}\sqrt{N} + \mathrm{poly}(\lambda)$.*

   Alternatively, for any $\ell = \omega(1)$, the probe complexity of the server can be improved to $N \cdot \ell / \log N + \mathrm{poly}(\lambda)$, at the expense of server storage $O(N^{1+\varepsilon}) + \mathrm{poly}(\lambda)$.

3. *Under the split-LSN conjecture (Conj. 1.2), there exists an sk-PIR protocol with storage* $(1+\varepsilon)N + \mathrm{poly}(\lambda)$ *and communication* $O(N^{\frac{1}{2}+\varepsilon}) + \mathrm{poly}(\lambda)$*, where the server has Boolean circuit size* $(4+\varepsilon)N + \mathrm{poly}(\lambda)$*.*

   The probe complexity of the server is $(3/4+\varepsilon)\cdot N$*, and can be improved to* $O(N/\log N)$ *at the expense of server storage and Boolean circuit size* $O(N^{1+\varepsilon}) + \mathrm{poly}(\lambda)$*.*

Proof. We begin by showing the correctness of the base protocol, assuming the noise rate of the underlying LSN assumption is $\mu = 0$ (as in the split-LSN based instance). That is, $0^n \in E$ with probability 1 and there always exists $\mathbf{d}$ such that $\mathbf{Q}^\intercal\mathbf{d} = \mathbf{c} + \mathbf{v}_{i_1}$ since $\mathbf{c}+\mathbf{v}_{i_1} \in \mathbf{c}+\mathbf{v}_{i_1} + E \subseteq \mathbf{Span}(\mathbf{Q})$. Correctness immediately follows by

$$y = (\mathsf{ans} - \mathbf{R}\mathbf{Q}^\intercal)\mathbf{d} = (\widehat{X}\mathbf{Q}^\intercal - \mathbf{R}\mathbf{Q}^\intercal)\mathbf{d} = X\mathbf{G}\mathbf{Q}^\intercal\mathbf{d} = X\mathbf{G}(\mathbf{c}+\mathbf{v}_{i_1}) = X\mathbf{G}\mathbf{v}_{i_1},$$

which is the i_1^{th} column in X by the definition of $\mathbf{v}_{i_1}$.

All sk-PIR protocols in Theorem 5.1 follow the same general recipe of instantiating the protocol from Fig. 2.

Denote by $\rho = k/n$ the rate in the assumed $(k, n, \mathcal{D})$-LSN. We choose security parameter $\lambda' \geq \lambda$ for the LSN assumption (to give as input to $\mathcal{D}$) that satisfies $n(\lambda') - k(\lambda') \geq N_1$ by $k := k(\lambda') = \lceil \rho N_1/(1-\rho)\rceil$ and $n := n(\lambda') = \lfloor N_1/(1-\rho)\rfloor$. (If N_1 is not large enough to allow such a choice, we set security parameter λ for the LSN and obtain a protocol with complexity polynomial in λ.)

By inspection, the protocol has storage $\widehat{N} = N_0 n = N/(1-\rho)$ and communication $q + rN_0$, where q is the description size of $\langle \mathbf{v} + E\rangle$ and r is the rank of $\mathbf{Span}(\langle \mathbf{v}+E\rangle)$. Security of the protocol follows by Lemma 5.1.

By choosing $N_0 = N_1 = \sqrt{N}$, the above already implies the first split-LSN-based instance in 3 in the theorem, where $\mu = 0$, ρ is any positive constant, $r = O(n^\delta)$ for arbitrarily small $\delta > 0$, and $q = O(n)$. The boolean circuit size of the server is $(4+\varepsilon)N$ since the computation for every database row consists of computing $2s$ inner products between (n/s)-length vectors, which can be performed with a circuit of size $4n$. In the cell probe model, the server looks at $(3/4+\varepsilon)N$ database bits in average: a bit is accessed when at least one of the two random vectors multiplied with the corresponding database chunk has 1 at the bit's location, and this occurs with probability 3/4.

When the LSN noise rate is $\mu > 0$, a simple strategy to correct errors is repetition. We let the client send $\tilde{\lambda} = \omega(\log\lambda/\log(1/\mu))$ i.i.d. queries to the server instead of a single one, and the server replies to all of the queries. Correctness now holds except with probability $\mu^{\tilde{\lambda}} = \lambda^{-\omega(1)}$. Negligible noise error can be turned into zero with negligible cost in security: whenever a query is noisy, the client makes a "plaintext query" (note the client in our protocol can tell whether a query is noisy before sending it, independently of the server's response). This gives the instance in 1 and the first in 2. The former is based on the reduction from LPN to sparse-noise LSN in Proposition 4.3 – recall the parameter ε of sparse-noise LSN (see Definition 4.3) corresponds to noise rate $\mu \approx \varepsilon(n-k)$ (see the discussion in (i) for more details). With $\varepsilon = 1/k^\gamma$, we choose $n = k + k^{\gamma'}$ for $\gamma' < \gamma$, to obtain $\mu = o(1)$ and $\tilde{\lambda} = \omega(\log\lambda)$, and $N_0 = N^{1/(1+\gamma')}$ and $N_1 = N^{\gamma'/(1+\gamma')}$. For the LSN-based instance we choose $N_0 = N_1 = \sqrt{N}$. We have ρ to be an arbitrarily small positive constant and $\mu = 1 - o(1)$. Correcting the noise requires $\tilde{\lambda} = \omega(\log\lambda/\log(1/\mu))$ repetitions, which can still be made any $\omega(\log\lambda)$ by a sufficiently small choice of μ.

To obtain the alternative instances in 2 and 3 with sublinear probe complexity, we observe that server computation consists of merely computing linear functions over the database and use a generic pre-processing technique for such computation. In the LSN-based instance, the server computes a parity function over the rows of the database and, in the split-LSN instance, it computes parity functions over (n/s)-length blocks of the rows. We hereby focus on the former, but the pre-processing works similarly for the latter when applied on the blocks individually.

Given an encoded database row $\widehat{x} \in \mathbb{F}^n$, the server additionally applies the following pre-processing. Divide $\widehat{x}$ into $B = n/\varepsilon \log n$ blocks $\widehat{x}_1, \ldots, \widehat{x}_B$ of length $L = \varepsilon \log n$ each. (W.l.o.g. we assume n is a power of 2.) For every $j \in [B]$, instead of storing $\widehat{x}_j$, the server stores the evaluation of all possible parity functions over $\widehat{x}_j$. That is, for every $\mathbf{q} \in \mathbb{F}_2^L$, he stores $y_j(\mathbf{q}) = \mathbf{q}^\intercal \widehat{x}_j \in \mathbb{F}_2$. This results in $B \cdot 2^L = O((n/\log n) \cdot n^\varepsilon)$ field elements in total. When answering a query $\mathbf{Q}$, the server divides each row $\mathbf{q}$ in $\mathbf{Q}$ into B functions $\mathbf{q}_1, \ldots, \mathbf{q}_B \in \mathbb{F}^L$, and computes the sum $\sum_j y_j(\mathbf{q}_j) = \widehat{x}\mathbf{q}^\intercal$. The size of this sum is at most $B = O(n/\log n)$, therefore, the probe complexity of one evaluation is $O(n/\log n)$. The probe complexity of evaluating over N_0 rows is then $O(\widehat{N}/\log \widehat{N})$.

While the above directly gives us a split-LSN-based sk-PIR with sublinear probe complexity, we need to be careful with the LSN-instance. Recall, to correct errors we let the server apply $\omega(\log \lambda)$ queries to each row, whereas pre-processing is able to save us only a factor of $O(\log N) = O(\log \lambda)$. Instead, we use a protocol where errors are handled by error-correction codes.

We let the client encode each of the database columns using, say, a Reed-Solomon code with dimension $\tilde{\lambda}$ and length $\ell\tilde{\lambda}$ over a field $\mathbb{F}$ of size $2^{N_0/\tilde{\lambda}}$, where $\ell = 1/(1-\mu) + \delta$ for a constant $\delta > 0$. Recall such a code can correct up to $(\ell - 1)\tilde{\lambda}$ erasures. The client divides each column into $\tilde{\lambda}$ field elements (each represented by a block of $N_0/\tilde{\lambda}$ bits) and encodes the column to obtain an encoded column that is ℓ times larger. The protocol is then invoked with the new larger database. To retrieve an element, the client sends $\tilde{\lambda}$ i.i.d. queries, and each is used over $N_0/\tilde{\lambda}$ rows corresponding to one field element. Overall, the server replies with ℓN_0 answers from which the client recovers ℓN_0 bits representing $\ell\tilde{\lambda}$ field elements that make a noisy codeword, which he can correct by erasure decoding. By Chernoff, the probability that more than $(\ell - 1)\tilde{\lambda} = (\mu/(1-\mu) + \delta)\tilde{\lambda} > (\mu + \delta)\tilde{\lambda}$ erasures occur is $e^{-\Omega(\tilde{\lambda})}$, which is negligible in λ. □

6 Improving Communication via Folding

Our folded sk-PIR protocol is presented in Fig. 3. It can be seen as a generalization of the the base protocol where the database is encoded using the tensor product of $t \geq 1$ random codes. (The base scheme is the special case with $t = 1$.)

Note that a client query in the folded protocol consists of t i.i.d. base-protocol queries. Hence, we may argue security along similar lines.

Lemma 6.1 (Folded Protocol Security)**.** *The sk-PIR from Fig. 3 is secure under* $(k, n, \mathcal{D})$-LSN*.*

Proof. Once again, using the security of the PRF, we first switch to a hybrid protocol where the PRF is replaced by a uniformly random function. In the hybrid protocol, the codes $C_1, \ldots, C_t$ are uniformly random and so is the mask $\mathbf{R}$. Since the database encoding is now uniform given the client queries (the queries are independent in $\mathbf{R}$), we consider an equally hard experiment where the adversary is not given any database encoding but only client queries of the form $(\mathbf{Q}_1, \ldots, \mathbf{Q}_t)$. By a standard hybrid argument involving t hybrids, we replace the distribution of $\mathbf{Q}_j = \langle \mathbf{Span}(\mathbf{c}_j + \mathbf{v}_{i_j} + E_j) \rangle$, for $j = 1, \ldots, t$, by a distribution $\mathbf{Q}'_j = \langle \mathbf{Span}(\mathbf{r}_j + E_j) \rangle$ for a uniformly random $\mathbf{r}_j$. The hybrids are computationally indistinguishable under the given LSN assumption due to Eq. (1). The last hybrid does not reveal any information about the vectors $\mathbf{v}_{i_j}$ and, therefore, about the client's inputs. Hence the proof is complete. □

We instantiate the folded protocol using LPN and LSN assumptions in the following theorem, and summarize the results in Table 2. Communication in the folded protocol can be made an

Parameters:

- Functions $t := t(N)$, $N_0 := N_0(N)$ and $N_1 := N_1(N)$ such that $N_0 N_1^t = N$.
  // We view a database $x \in \{0,1\}^N$ as $X \in \mathbb{F}_2^{N_0 \times N_1 \times \cdots \times N_1}$ and $i \in [N]$ as $(i_0, i_1, \ldots, i_t) \in [N_0] \times [N_1]^t$.
- Code dimension $k := k(\lambda)$.
- A pseudorandom function $\mathsf{PRF} : \{0,1\}^\lambda \times \{0,1\}^\lambda \to \{0,1\}^\lambda$.

The Protocol:

- $\mathsf{G}(1^\lambda, 1^N)$: Output a uniform PRF key $\mathsf{sk} \leftarrow \{0,1\}^\lambda$.
- $\mathsf{E}(\mathsf{sk},\ X \in \mathbb{F}^{M \times N_0^t})$: Use the outputs of $\mathsf{PRF}(\mathsf{sk}, \cdot)$ as randomness in the following:
  1. For any $j = 1, \ldots, t$, sample $(C_j, \mathcal{E}_j) \leftarrow \mathcal{D}(1^\lambda, 1^k, 1^n)$ for $n = k + N_1$.
     - Let $\mathbf{G}_j \in \mathbb{F}_2^{N_1 \times n}$ be a generator matrix of $D_j = C_j^\perp$.
     - For any $i \in [N_1]$, let $\mathbf{v}_{j,i} \in \mathbb{F}_2^n$ be a vector for which $\mathbf{G}_j \mathbf{v}_{j,i} \in \mathbb{F}_2^{N_1}$ is the i^{th} unit vector.
  2. Sample a uniformly random mask $\mathbf{R} \leftarrow \mathbb{F}_2^{N_0 \times n \times \cdots \times n}$.

  Output the following database encoding
  $$\widehat{X} = X(\mathbf{G}_1 \otimes \cdots \otimes \mathbf{G}_t) + \mathbf{R} \in \mathbb{F}^{N_0 \times n \times \cdots \times n},$$
- $\mathsf{Q}(\mathsf{sk}, (i_0, \ldots, i_t))$: For $j = 1, \ldots, t$, sample $\mathbf{c}_j \leftarrow C_j$ and $E_j \leftarrow \mathcal{E}_j$. Let $\mathbf{Q}_j = \langle \mathbf{Span}(\mathbf{c}_j + \mathbf{v}_{i_j} + E_j) \rangle \in \mathbb{F}^{r \times n}$ and output
  $$\mathsf{que} = (\mathbf{Q}_1, \ldots, \mathbf{Q}_t) \qquad\qquad \mathsf{aux} = ((\mathbf{Q}_1, \ldots, \mathbf{Q}_t), (\mathbf{d}_1, \ldots, \mathbf{d}_t)),$$
  where $\mathbf{d}_j \in \mathbb{F}_2^r$ is the vector satisfying $\mathbf{Q}_j^\intercal \mathbf{d}_j = \mathbf{c}_j + \mathbf{v}_{j,i_j}$ if exists or $\perp$ if it does not exist.
- $\mathsf{A}(\widehat{X}, \mathsf{que})$: Output $\mathsf{ans} = \widehat{X}(\mathbf{Q}_1 \otimes \cdots \otimes \mathbf{Q}_t)^\intercal \in \mathbb{F}_2^{N_0 \times r \times \cdots \times r}$.
- $\mathsf{D}(\mathsf{sk}, (i_0, \ldots, i_t), \mathsf{ans}, \mathsf{aux})$: Output the i_0^{th} bit of $y = (\mathsf{ans} - \mathbf{R}(\mathbf{Q}_1 \otimes \cdots \otimes \mathbf{Q}_t)^\intercal)(\mathbf{d}_1 \otimes \cdots \otimes \mathbf{d}_t) \in \mathbb{F}^{N_0}$ or $\perp$.

Figure 3: Our Folded sk-PIR Protocol under $\mathcal{D}$-LSN.

arbitrarily small polynomial in N by choosing the folding parameter t to be a sufficiently large constant. Folding with $t = \omega(1)$ can give an even smaller communication of $N^{o(1)}$, and in one instance even sublinear probe complexity, under quasi-polynomial hardness. (When $t = \omega(1)$, the LSN dimension becomes proportional to $N^{1/t} = N^{o(1)}$, which requires quasi-polynomial hardness when N is considered to be $\mathrm{poly}(\lambda)$.) The folded protocol too offers a trade-off between server storage and computation via pre-processing, which is even better than in the base protocol.

Theorem 6.1 (Folded sk-PIR from LSN Assumptions)**.** *The following holds for any constant $t \geq 1$, any $\varepsilon > 0$ and any $\tilde{\lambda} = \omega(\log \lambda)$:*

1. *Under the $(k, 1/k^\gamma)$-LPN assumption, for any $0 < \gamma' < \gamma$, there exists sk-PIR with storage $O(N^{(t+1)/(\gamma' t+1)}) + \mathrm{poly}(\lambda)$ and communication $O(N^{1/(\gamma' t+1)})$.*

2. *Under the LSN conjecture (Conj. 1.1), there exists sk-PIR with storage $(1+\varepsilon)N + \mathrm{poly}(\lambda)$ and communication $O(\tilde{\lambda} N^{1/(t+1)} + \mathrm{poly}(\lambda))$.*

   Alternatively, for any $\ell = \omega(1)$, the probe complexity of the server can be improved to $N \cdot \ell / \log^t N + \mathrm{poly}(\lambda)$, at the expense of server storage $O(N^{1+\varepsilon}) + \mathrm{poly}(\lambda)$.

3. *Under the split-LSN conjecture (Conj. 1.2), there exists sk-PIR with storage $(1+\varepsilon)N + \mathrm{poly}(\lambda)$ and communication $O(N^{1/(t+1)+\varepsilon}) + \mathrm{poly}(\lambda)$, where the server has Boolean circuit size $(4 + \varepsilon)N + \mathrm{poly}(\lambda)$.*

   The probe complexity of the server is $(3/4+\varepsilon)^t N$ and can be improved to $O(N/\log^t N)$ at the expense of server storage and Boolean circuit size $O(N^{1+\varepsilon})$.

# in Thm. 6.1	Conjecture	Communication	Server		
			Storage	Boolean circuit size	Probes
1	High-noise LPN	N^{ε}	$N^{2+\varepsilon}$	-	-
	Mid-noise LPN	N^{ε}	$N^{1+\varepsilon}$	-	-
2	LSN (Conj. 1.1)	N^{ε}	$(1+\varepsilon)N$	-	-
		N^{ε}	$N^{1+\varepsilon}$	-	$\omega(1) \cdot N / \log^{1/\varepsilon} N$
3	Split-LSN (Conj. 1.2)	N^{ε}	$(1+\varepsilon)N$	$(4+\varepsilon)N$	εN
Rem. 6.1	Quasi-poly Split-LSN	$N^{o(1)}$	$(1+\varepsilon)N$	$(4+\varepsilon)N$	$o(N)$

Table 2: Instantiations of our folded protocol from Figure 3 under different LPN (Definition 3.4) and LSN assumptions.We consider N to be also a security parameter, namely we require that any poly(N)-time adversary has advantage at most negl(N). We refer the reader to Theorem 6.1 for precise details.

We hereby state the instantiation under quasi-polynomial split-LSN, which achieves sublinear probe complexity with small storage.

Remark 6.1 (sk-PIR under quasi-polynomial split-LSN)**.** *By instantiating the protocol from Item 3 in Theorem 6.1 with a super-constant $t = \omega(1)$, we obtain sk-PIR under the quasi-polynomial hardness of split-LSN (with $\rho = o(1)$ and otherwise parameters similar to Conjecture 1.2) with the same server storage and circuit size, communication $N^{o(1)} + \text{poly}(\lambda)$ and probe complexity of $o(N)$.*

Using a generic compiler by Boyle et al. [BIPW17], all of the protocols above can be converted to standard (public-key) PIR with trusted setup by additionally assuming the existence of *virtual black-box (VBB) obfuscation*. In a model with trusted setup, it is assumed that the database encoding is generated by a trusted party, together with a public key that is made available to any client. Alternatively, trust can be eliminated by assuming a common-reference string (CRS) and universal samplers [HJK$^+$16]. In a nutshell, the construction (implicit in [BIPW17, Theorem 5.1]) encodes the database under an sk-PIR protocol, and uses VBB to obfuscate two programs: (i) A "query program" that, on input client query $i \in [N]$ and randomness, generates an sk-PIR query corresponding to i. (ii) A "decoding program" which, on input an sk-PIR query and its response from the server, decodes the query and outputs the underlying database bit value. In addition, to prevent trivial attacks, symmetric-key cryptography (e.g. MACs) is used to enforce that an input to the decoding program was generated using the query program and a given i. Given these programs as the public key, the protocol is straight-forward: the client uses the query program to query the database and decodes the server's response using the decoding program.

Remark 6.2. *All statements of Theorem 6.1 and Remark 6.1 hold with respect to public-key PIR with trusted setup in the ideal obfuscation model.*

We proceed to prove Theorem 6.1.

Proof. Let us first prove correctness. The client is able to retrieve the requested element whenever

$\mathbf{d}_j$ exists for all j. This is because, in such a case,

$$\begin{aligned}
y &= (\mathsf{ans} - \mathbf{R}(\mathbf{Q}_1 \otimes \cdots \otimes \mathbf{Q}_t)^\intercal)(\mathbf{d}_1 \otimes \cdots \otimes \mathbf{d}_t)\\
&= (\widehat{X} - \mathbf{R})(\mathbf{Q}_1 \otimes \cdots \otimes \mathbf{Q}_t)^\intercal(\mathbf{d}_1 \otimes \cdots \otimes \mathbf{d}_t)\\
&= X(\mathbf{G}_1 \otimes \cdots \otimes \mathbf{G}_t)(\mathbf{Q}_1^\intercal \mathbf{d}_1 \otimes \cdots \otimes \mathbf{Q}_t^\intercal \mathbf{d}_t)\\
&= X(\mathbf{G}_1(\mathbf{c}_1 + \mathbf{v}_{1,i_1}) \otimes \cdots \otimes \mathbf{G}_t(\mathbf{c}_t + \mathbf{v}_{t,i_t}))\\
&= X(\mathbf{G}_1\mathbf{v}_{1,i_1} \otimes \cdots \otimes \mathbf{G}_t\mathbf{v}_{t,i_t})\\
&= X_{i_1,\ldots,i_t}.
\end{aligned}$$

where $X_{i_1,\ldots,i_t}$ denotes the $\mathbb{F}_2^{N_0}$-column consisting of all bits in X at coordinates of the form $(*, i_1, \ldots, i_t)$ and the last equality follows from the choice of the $\mathbf{v}_{j,i_j}$'s. The requested bit is then in the i_0^{th} location of this column.

Denote $\rho = k/n$. We choose security parameter $\lambda' \geq \lambda$ for the LSN assumption (to give as input to $\mathcal{D}$) that satisfies $n(\lambda') - k(\lambda') \geq N_1$ by $k := k(\lambda') = \lceil \rho N_1/(1-\rho) \rceil$ and $n := n(\lambda') = N_1/(1-\rho)$. (If N_1 is not large enough to allow such a choice, we set security parameter λ for the LSN and obtain is protocol with complexity polynomial in λ). The database encoding in the folded protocol has size $\widehat{N} = N/(1-\rho)^t$ and communication $tq + N_0 r^t$, where r and q are the rank and, resp., the description size of an LSN sample $\langle \mathbf{c} + E \rangle$.

By choosing $N_0 = N_1 = N^{1/(t+1)}$, we derive the split-LSN based instance of the protocol (3), where the noise rate is $\mu = 0$ (all $\mathbf{d}_j$ always exist), the rate ρ is such that $1/(1-\rho)^t < 1 + \varepsilon$, the rank is $r = 2s = O(N^\delta)$ for arbitrarily small $\delta > 0$ and the description size is $q = O(N^{1/(t+1)})$. We defer the analysis of probe complexity and Boolean circuit size to the end of this proof.

For the LPN-based instance, with LPN noise $\varepsilon = 1/k^\gamma$, we again set $n = k + k^{\gamma'}$ for $\gamma' < \gamma$ and get $\mu = o(1)$. We choose $N_0 = N^{1/(1+\gamma' t)}$ and $N_1 = N^{\gamma'/(1+\gamma' t)}$ to balance communication. For the LSN-based instance, we set $N_0 = N_1 = N^{1/(t+1)}$, $\rho > 0$ to be sufficiently small as above and $\mu = 1 - o(1)$. In these noisy instantiations with $\mu > 0$, all $\mathbf{d}_j$ exist with probability $(1-\mu)^t$ and, therefore, we have a noisy query with probability $1 - (1-\mu)^t$. Similarly to the base protocol, we apply either repetition or erasure-correction to each of the N_1^t database columns before we encode it under the sk-PIR. The details, including choice of parameters, are identical to our noise-handling strategy for the base scheme, which we fully describe in the proof of Theorem 5.1, up to substituting the noise rate with $(1-\mu)^t$.

The alternative protocols with sublinear probe complexity are obtained by the same preprocessing technique that we applied to the base protocol (see the proof of Theorem 5.1). Here, however, an even smaller probe complexity of $O(N/\log^t N)$ is enabled by the fact that the server applies low-rank linear functions over the database rows, i.e. inner-products with vectors of the form $(\mathbf{q}_1 \otimes \cdots \otimes \mathbf{q}_t) \in \mathbb{F}^{n^t}$. Consequently, when dividing each row in the database into B^t blocks of length L^t each, where $L = \varepsilon \log n$ and $B = n/L$, the number of all possible linear functions that the server may evaluate over each block is at most $|\mathbb{F}|^{tL} = O(n^{\varepsilon t})$ (instead of all 2^{L^t}). The probe complexity for a single row is then $B^t = O(n^t/\log^t n)$.

To see why the probe complexity of the server in the split-LSN based protocol is $(3/4+\varepsilon)^t N$, note that a location in the database is probed if it appears with a non-zero coefficient in the linear function that the server performs over the database. The linear function is a t-fold tensor of smaller functions, each applied along one of the dimensions in the tensor representation of the database. For a location to be probed, then, it must appear in all of the t linear functions (otherwise its coefficient in the tensor is zero). For a location to appear in one of the linear functions it must appear in at least one of the two vectors that are used to hide the query codeword over the corresponding (n/s)-bit chunk (recall Definition 1.2). This occurs with probability $1 - (1/2)^2 = 3/4$ for each location in any of the t dimensions. Hence, a location is probed with probability $(3/4)^t$ and there are $N/(1-\rho)^t$ locations.

Lastly, we analyze the Boolean circuit size of the server. To answer a query, the server evaluates $(\widehat{X}_i, (\mathbf{Q}_1 \otimes \cdots \otimes \mathbf{Q}_t)) \mapsto \widehat{X}_i(\mathbf{Q}_1 \otimes \cdots \otimes \mathbf{Q}_t)^\intercal$ for each of the N_0 database rows $\widehat{X}_i \in \mathbb{F}_2^{n \times \cdots \times n}$. Consider the following circuit that evaluates such a function by "folding" the tensor $\widehat{X}_i$ one dimension at a time: First, it computes $\widehat{X}_i(\mathbf{Q}_1 \otimes I \otimes \cdots \otimes I)^\intercal$ to obtain an output $\widehat{X}_i^1 \in \mathbb{F}_2^{r \times n \times \cdots \times n}$ of length rn^{t-1}. This can be implemented by n^{t-1} copies of the circuit computing $x \mapsto x\mathbf{Q}_1^\intercal$ over inputs x of length n (applied over the "lines" in $\widehat{X}_i$ parallel to the first axis). This shrinks the first dimension of $\widehat{X}_i$ from length n to length r. Next, the circuit computes $\widehat{X}_i^1(I \otimes \mathbf{Q}_2 \otimes I \otimes \cdots \otimes I)^\intercal$ which corresponds to $r \cdot n^{t-2}$ copies of the circuit computing $x \mapsto x\mathbf{Q}_2^\intercal$ over x of length n, to obtain $\widehat{X}_i^2$ of length $r^2 n^{t-2}$. The computation completes after t such iterations, resulting in an answer of size r^t. Overall, letting S denote circuit size of the computation $x \mapsto x\mathbf{Q}_j^\intercal$, the Boolean circuit size for computing the answer for one row of the database is bounded by $\sum_{j=1}^{t} n^{t-j} r^{j-1} S = (1+o(1))n^{t-1}S$ and $(1+o(1))N_0 n^{t-1} S$ for all rows in total. In the split-LSN-based instance where $S = 4n$ this is $(1+o(1))4nN_0 < (4+\varepsilon)N$. □

7 Improving Client Runtime

The client in the folded protocols from Theorem 6.1 can always be implemented with (per-query) online time sublinear in N. Specifically, in any instantiation of the theorem with $t > 1$, where communication is $O(N^\varepsilon)$, the client runs in time $\tilde{O}(N^{1-\varepsilon})$ (here and in the following, $\tilde{O}()$ hides polylogaritymic factors, and we ignore additive poly(λ) terms). To generate a query, the client samples t uniformly random codewords, each from an $N_1 \times n$ generator matrix, taking time $O(tN_1 n) = O(N^{(2+\varepsilon)/(t+1)})$. (Recall that n is $O(N_1^{1/\gamma})$ under LPN for $\gamma < 1$ and is $O(N_1)$ in the other LSN-based protocols.) To decode an answer from the server, the client uncomputes the effect of the mask $\mathbf{R}$ by performing an inner product over inputs of length $n^t = O(N^{t/(t+1)})$.

In this section, we demonstrate how to reduce even further the client runtime in our protocols via two different strategies. As a result, we get the following:

- Using LPN-based optimization techniques, we reduce the client runtime in the LPN-based folded protocol. We obtain an sk-PIR with communication and client runtime both $O(N^{1/3+\varepsilon})$.
- We use standard PIR to generically bootstrap the client in all of our protocols. We obtain sk-PIR under LSN or split-LSN with communication and client runtime $O(N^\varepsilon)$, by additionally assuming the existence of standard PIR with small communication and efficient client. Importantly, the server circuit size, probe complexity, and storage in these protocols are preserved by the transformation, as long as they are all $O(N^{1+\varepsilon'})$ in the PIR protocol, for an arbitrarily small $\varepsilon' > 0$. Such PIR protocols are known under most standard assumptions known to imply public-key cryptography, including DDH, QRA and LWE.

7.1 Cubic-Root Client Runtime under LPN

Consider the LPN-based folded protocol with $t = 3$. By the discussion above, each of the client's operations: generating codewords for the query and uncomputing the mask for decoding, take time $O(N^{2/3+\varepsilon})$. We reduce the complexity of each of these operations to $O(N^{1/3+\varepsilon})$.

To make the query generation efficient, we rely on a sparse-noise LSN variant (see Definition 4.3) where the hidden code has a sparse generator matrix rather than uniform. Note generating a codeword from such a matrix is much more efficient compared to a random code. We show that this variant is still implied by LPN via a reduction similar to Proposition 4.3. Our reduction is, in fact, from an LPN variant where the secret is sparse – it is sampled from the error distribution. This variant is known to be implied by standard LPN [ACPS09].

Lemma 7.1 (LPN with Sparse Secrets [ACPS09])**.** *The* (k, ε)-LPN *over* $\mathbb{F}$ *implies that for any polynomial* $m := m(\lambda)$*,*

$$(\mathbf{a}_i, \mathbf{a}_i^\intercal \mathbf{s} + r_i)_{i \in [m]} \approx_c (\mathbf{a}_i, u_i)_{i \in [m]},$$

where $\mathbf{s} \leftarrow \mathsf{Ber}_{\mathbb{F}}^k(\varepsilon)$ *and, for any* i*,* $\mathbf{a}_i \leftarrow \mathbb{F}^k$*,* $r_i \leftarrow \mathsf{Ber}_{\mathbb{F}}(\varepsilon)$ *and* $u_i \leftarrow \mathbb{F}$*.*

Based on the above, we obtain the following reduction from LPN to sparse-noise LSN where the hidden code has a sparse generator matrix.

Lemma 7.2 (LSN with Sparse Generator Matrix from LPN)**.** *The* (k, ε)-LPN *assumption implies the* $(k, n, \mathcal{D})$-LSN *assumption where* $\mathcal{D}(1^\lambda, 1^k, 1^n)$ *samples* $(C, \mathcal{E})$ *as follows:*

- *C is the row-span of the generator matrix $(I, \mathbf{S}) \in \mathbb{F}^{k \times n}$ where $\mathbf{S} \leftarrow \mathsf{Ber}(\varepsilon)^{k \times (n-k)}$.*
- *$\mathcal{E}$ is a fixed sampler that outputs $E = \{\mathbf{r}\}$ where $\mathbf{r}_j = 0$ for $j = 1, \ldots, k$ and $\mathbf{r}_j \leftarrow \mathsf{Ber}(\varepsilon)$ for $j = k+1, \ldots, n$.*

Proof. By Lemma 7.1, it suffices to show the implication under an LPN variant where the bits in the secret $\mathbf{s}$ are sampled from the error distribution $\mathsf{Ber}(\varepsilon)$. By a hybrid argument identical to that in the proof of Lemma 4.2 (see Appendix A), such LPN implies its matrix analog which asserts that, for any polynomial $t := t(\lambda)$, polynomially many samples of the form $\mathbf{a}_i, \mathbf{a}_i^\intercal \mathbf{S} + \mathbf{e}_i$ are pseudorandom, where $\mathbf{S} \leftarrow \mathsf{Ber}(\varepsilon)^{k \times t}$ and, for any i, $\mathbf{a}_i \leftarrow \mathbb{F}^k$, and $\mathbf{e}_i \leftarrow \mathsf{Ber}^n(\varepsilon)$. The proof is complete since such samples for $t = n - k$ are equivalent to samples from the LSN in the lemma: They can be rewritten as $\mathbf{a}_i(I, \mathbf{S}) + \mathbf{r}_i$, where $\mathbf{r}_i \leftarrow \mathcal{E}$. □

To optimize the time complexity of uncomputing the mask, we rely on a tool developed by two recent works [BN25, VZ25]. These works show that, under LPN, one can sample a pseudorandom $n \times n$ matrix and, using a trapdoor, to multiply any vector by the matrix in time almost linear in n.

Lemma 7.3 (Trapdoored Matrices, implicit in [BN25, VZ25])**.** *Assuming* $(k, 1/k^\gamma)$-LPN *holds for any* $\gamma > 0$*, there exists, for any* $\varepsilon > 0$ *and polynomial* $n := n(\lambda)$*, an efficient sampler that samples a pseudorandom matrix* $\mathbf{M} \in \mathbb{F}^{n \times n}$ *such that, given the randomness used to sample* $\mathbf{M}$*, matrix-vector multiplication* $\mathbf{M}\mathbf{v}$ *takes time* $O(n^{1+\varepsilon})$ *for any* $\mathbf{v}$*.*

We note that the above statement is slightly different than the statements in the original works (Theorem 1 in [BN25] and 3.1 in [VZ25]), where multiplication is given a trapdoor. It is by inspection, though, that the trapdoor can be re-computed from the randomness in time less than what it takes to compute the multiplication.

The above allows the client to sample the mask over each database row as a pseudorandom trapdoored matrix, rather than a uniform one, while preserving security and reducing decoding time.

By combining the above two modifications to the LPN-based folded protocol, we obtain the following sk-PIR.

Corollary 7.1. *Assuming* $(k, 1/k^\gamma)$-LPN *holds for any* $\gamma > 0$*, there exists, for any* $\varepsilon > 0$*, an sk-PIR protocol with storage* $N^{1+\varepsilon} + \mathrm{poly}(\lambda)$ *and communication* $N^{\frac{1}{3}+\varepsilon} + \mathrm{poly}(\lambda)$*, where the client runs in time* $N^{\frac{1}{3}+\varepsilon} + \mathrm{poly}(\lambda)$ *per query.*

Proof. The protocol is obtained by optimizing client runtime in the LPN-based folded sk-PIR from Theorem 6.1 (instance 1) with $t = 2$, using Lemmas 7.2 and 7.3.

First, by Lemma 7.2, we may replace the sparse-noise LSN sampler, which underlies the protocol from Theorem 6.1, by its variant with sparse generator from the lemma, without affecting the security of the protocol. Specifically, in the database encoding, the client generates a generator matrix $\mathbf{H}_j \in \mathbb{F}^{k\times n}$ for the code C_j^q (which is parity-check for $\mathbf{G}_j$) as follows: Using the PRF, it samples the locations where $\mathbf{H}_j$ contains 1 in the non-systematic part (denoted $\mathbf{S}$ in Lemma 7.2), which are $k^{1-\gamma}n = O(N^{\frac{1}{3}+\varepsilon})$ in total. When generating a query, given the PRF values, the client can sample a uniformly random codeword in the span of $\mathbf{H}_j$ in time $O(N^{\frac{1}{3}+\varepsilon})$.

Second, we replace every row in the mask $\mathbf{R} \in \mathbb{F}^{N_0\times(n\times n)}$ sampled by the client to be sampled as a trapdoored $n \times n$ matrix by the sampler from Lemma 7.3. Security still follows under LPN by the psuedorandomness of the mask. In the decoding, to uncompute the mask for the i_0^{th} answer bit (which is the only one interesting to the client), the client computes $\mathbf{R}_{i_0}(\mathbf{Q}_1 \otimes \mathbf{Q}_2)^\intercal = \mathbf{Q}_1\mathbf{R}_{i_0}\mathbf{Q}_2^\intercal$, where $\mathbf{R}_{i_0} \in \mathbb{F}^{n\times n}$ is the mask at row i_0 and $\mathbf{Q}_1, \mathbf{Q}_2 \in \mathbb{F}^{r\times n}$. This involves r vector multiplications with $\mathbf{R}_{i_0}$ and additional r inner-products of length n, giving time complexity $O(rn^{1+\varepsilon}) = O(N^{\frac{1}{3}+\varepsilon})$. □

7.2 Composing with Standard PIR

Lastly, we use plain-model PIR as a tool to bootstrap any of our small-communication protocols from Theorem 6.1 to have client runtime of N^ε. In fact, it is sufficient to apply the bootstrapping over the base protocol (recall it is a special case of *Theorem* 6.1).

Proposition 7.1. *Assume the existence of a (plain model) PIR protocol with communication N^ε, for arbitrarily small $\varepsilon > 0$, and storage, probe complexity and server Boolean circuit size all at most $N^{1+\varepsilon}$. Then, the conclusions of Theorem 6.1 hold with protocols that have client runtime complexity of at most N^ε per query.*

Proof. The instantiations are obtained by "folding" the base protocol from Fig. 2 with the given PIR. More concretely, starting with the base protocol, we choose $N_0 = N^{1-\varepsilon}$ and $N_1 = N^\varepsilon$. The client sends its query que with respect to an N^ε size database and, additionally, a (standard) PIR query que' corresponding to a location $i_0 \in [N_1]$. The server applies que over all rows, obtaining a column of N_1 "sk-PIR answers". It then applies que' over the column to obtain a "PIR answer", which it sends to the client. □

Acknowledgements. We thank Martin Albrecht, Fabrice Benhamouda, Anindya De, Itai Dinur, Shai Halevi, Swastik Kopparty, Hugo Krawczyk, Tal Rabin, and Jean-Pierre Tillich for helpful discussions.

References

[ACPS09] Benny Applebaum, David Cash, Chris Peikert, and Amit Sahai. Fast cryptographic primitives and circular-secure encryption based on hard learning problems. In Shai Halevi, editor, *Advances in Cryptology – CRYPTO 2009*, volume 5677 of *Lecture Notes in Computer Science*, pages 595–618, Santa Barbara, CA, USA, August 16–20, 2009. Springer Berlin Heidelberg, Germany. 12, 30

[AG11] Sanjeev Arora and Rong Ge. New algorithms for learning in presence of errors. In Luca Aceto, Monika Henzinger, and Jirí Sgall, editors, *Automata, Languages and Programming - 38th International Colloquium, ICALP 2011, Zurich, Switzerland, July 4-8, 2011, Proceedings, Part I*, volume 6755 of *Lecture Notes in Computer Science*, pages 403–415. Springer, 2011. 4, 18

[AHI$^+$17] Benny Applebaum, Naama Haramaty, Yuval Ishai, Eyal Kushilevitz, and Vinod Vaikuntanathan. Low-complexity cryptographic hash functions. In Christos H. Papadimitriou, editor, *8th Innovations in Theoretical Computer Science Conference, ITCS 2017, January 9-11, 2017, Berkeley, CA, USA*, volume 67 of *LIPIcs*, pages 7:1–7:31. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2017. 2, 7

[AIK07] Benny Applebaum, Yuval Ishai, and Eyal Kushilevitz. Cryptography with constant input locality. In *Annual International Cryptology Conference*, pages 92–110. Springer, 2007. 9, 16, 37

[Ajt10] Miklós Ajtai. Oblivious rams without cryptogrpahic assumptions. In Leonard J. Schulman, editor, *Proceedings of the 42nd ACM Symposium on Theory of Computing, STOC 2010, Cambridge, Massachusetts, USA, 5-8 June 2010*, pages 181–190. ACM, 2010. 6

[Ale03] Michael Alekhnovich. More on average case vs approximation complexity. In *44th Symposium on Foundations of Computer Science (FOCS 2003), 11-14 October 2003, Cambridge, MA, USA, Proceedings*, pages 298–307. IEEE Computer Society, 2003. 3, 15

[AM13] Benny Applebaum and Yoni Moses. Locally computable UOWHF with linear shrinkage. In Thomas Johansson and Phong Q. Nguyen, editors, *Advances in Cryptology – EUROCRYPT 2013*, volume 7881 of *Lecture Notes in Computer Science*, pages 486–502, Athens, Greece, May 26–30, 2013. Springer Berlin Heidelberg, Germany. 7

[AMR25] Damiano Abram, Giulio Malavolta, and Lawrence Roy. Trapdoor hash functions and PIR from low-noise LPN. Cryptology ePrint Archive, Paper 2025/416, 2025. 3, 15

[BCG$^+$19] Elette Boyle, Geoffroy Couteau, Niv Gilboa, Yuval Ishai, Lisa Kohl, and Peter Scholl. Efficient pseudorandom correlation generators: Silent OT extension and more. In Alexandra Boldyreva and Daniele Micciancio, editors, *Advances in Cryptology – CRYPTO 2019, Part III*, volume 11694 of *Lecture Notes in Computer Science*, pages 489–518, Santa Barbara, CA, USA, August 18–22, 2019. Springer, Cham, Switzerland. 16

[BCG$^+$23] Elette Boyle, Geoffroy Couteau, Niv Gilboa, Yuval Ishai, Lisa Kohl, Nicolas Resch, and Peter Scholl. Oblivious transfer with constant computational overhead. In Carmit Hazay and Martijn Stam, editors, *Advances in Cryptology - EUROCRYPT 2023 - 42nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Lyon, France, April 23-27, 2023, Proceedings, Part I*, volume 14004 of *Lecture Notes in Computer Science*, pages 271–302. Springer, 2023. 2, 7

[BFKL94] Avrim Blum, Merrick Furst, Michael Kearns, and Richard J. Lipton. Cryptographic primitives based on hard learning problems. In Douglas R. Stinson, editor, *Advances in Cryptology — CRYPTO' 93*, pages 278–291, Berlin, Heidelberg, 1994. Springer Berlin Heidelberg. 3, 14

[BHMW21] Elette Boyle, Justin Holmgren, Fermi Ma, and Mor Weiss. On the security of doubly efficient pir. *Cryptology ePrint Archive*, 2021. 1, 5

[BHW19] Elette Boyle, Justin Holmgren, and Mor Weiss. Permuted puzzles and cryptographic hardness. In Dennis Hofheinz and Alon Rosen, editors, *Theory of Cryptography - 17th International Conference, TCC 2019, Nuremberg, Germany, December 1-5, 2019, Proceedings, Part II*, volume 11892 of *Lecture Notes in Computer Science*, pages 465–493. Springer, 2019. 1, 5

[BIM00] Amos Beimel, Yuval Ishai, and Tal Malkin. Reducing the servers computation in private information retrieval: Pir with preprocessing. In *Advances in Cryptology—CRYPTO 2000: 20th Annual International Cryptology Conference Santa Barbara, California, USA, August 20–24, 2000 Proceedings 20*, pages 55–73. Springer, 2000. 1, 4

[BIPW17] Elette Boyle, Yuval Ishai, Rafael Pass, and Mary Wootters. Can we access a database both locally and privately? In *Theory of Cryptography: 15th International Conference, TCC 2017, Baltimore, MD, USA, November 12-15, 2017, Proceedings, Part II 15*, pages 662–693. Springer, 2017. 1, 2, 4, 5, 6, 7, 13, 18, 27

[BLVW19] Zvika Brakerski, Vadim Lyubashevsky, Vinod Vaikuntanathan, and Daniel Wichs. Worst-case hardness for lpn and cryptographic hashing via code smoothing. In *Advances in Cryptology – EUROCRYPT 2019: 38th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Darmstadt, Germany, May 19–23, 2019, Proceedings, Part III*, page 619–635, Berlin, Heidelberg, 2019. Springer-Verlag. 3, 15

[BN25] Mark Braverman and Stephen Newman. Sublinear-overhead secure linear algebra on a dishonest server. *CoRR*, abs/2502.13060, 2025. 6, 11, 30

[BW21] Keller Blackwell and Mary Wootters. A note on the permuted puzzles toy conjecture. *arXiv preprint arXiv:2108.07885*, 2021. 1, 5

[CDV21] Aidao Chen, Anindya De, and Aravindan Vijayaraghavan. Learning a mixture of two subspaces over finite fields. In Vitaly Feldman, Katrina Ligett, and Sivan Sabato, editors, *Algorithmic Learning Theory, 16-19 March 2021, Virtual Conference, Worldwide*, volume 132 of *Proceedings of Machine Learning Research*, pages 481–504. PMLR, 2021. 2, 3, 9, 15, 17, 20, 21

[CGKS95] Benny Chor, Oded Goldreich, Eyal Kushilevitz, and Madhu Sudan. Private information retrieval. In *36th Annual Symposium on Foundations of Computer Science, Milwaukee, Wisconsin, USA, 23-25 October 1995*, pages 41–50. IEEE Computer Society, 1995. 1, 8

[CHR17] Ran Canetti, Justin Holmgren, and Silas Richelson. Towards doubly efficient private information retrieval. In *Theory of Cryptography: 15th International Conference, TCC 2017, Baltimore, MD, USA, November 12-15, 2017, Proceedings, Part II 15*, pages 694–726. Springer, 2017. 1, 2, 5, 7, 13, 18

[CK20] Henry Corrigan-Gibbs and Dmitry Kogan. Private information retrieval with sublinear online time. In *Advances in Cryptology–EUROCRYPT 2020: 39th Annual Interna-*

tional Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, May 10–14, 2020, Proceedings, Part I 39, pages 44–75. Springer, 2020. 6

[CM19] Sitan Chen and Ankur Moitra. Beyond the low-degree algorithm: mixtures of subcubes and their applications. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2019, page 869–880, New York, NY, USA, 2019. Association for Computing Machinery. 15

[dCHI$^+$22] Leo de Castro, Carmit Hazay, Yuval Ishai, Vinod Vaikuntanathan, and Muthu Venkitasubramaniam. Asymptotically quasi-optimal cryptography. In Orr Dunkelman and Stefan Dziembowski, editors, *Advances in Cryptology – EUROCRYPT 2022, Part I*, volume 13275 of *Lecture Notes in Computer Science*, pages 303–334, Trondheim, Norway, May 30 – June 3, 2022. Springer, Cham, Switzerland. 7

[DMN11] Ivan Damgrard, Sigurd Meldgaard, and Jesper Buus Nielsen. Perfectly secure oblivious ram without random oracles. In *Theory of Cryptography Conference (TCC)*, volume 6597 of *Lecture Notes in Computer Science*, pages 144–163. Springer, 2011. 6

[DMO00] Giovanni Di Crescenzo, Tal Malkin, and Rafail Ostrovsky. Single database private information retrieval implies oblivious transfer. In Bart Preneel, editor, *Advances in Cryptology - EUROCRYPT 2000, International Conference on the Theory and Application of Cryptographic Techniques, Bruges, Belgium, May 14-18, 2000, Proceeding*, volume 1807 of *Lecture Notes in Computer Science*, pages 122–138. Springer, 2000. 1

[Döt14] Nico Döttling. *Cryptography based on the Hardness of Decoding*. PhD thesis, Karlsruhe Institute of Technology, 2014. 10, 21

[DP12] Ivan Damgrard and Sunoo Park. How practical is public-key encryption based on LPN and ring-LPN? Cryptology ePrint Archive, Paper 2012/699, 2012. 21

[FLY22] Zhiyuan Fan, Jiatu Li, and Tianqi Yang. The exact complexity of pseudorandom functions and the black-box natural proof barrier for bootstrapping results in computational complexity. In *STOC '22*, pages 962–975, 2022. 7

[FSO06] Jon Feldman, Rocco A. Servedio, and Ryan O'Donnell. Pac learning axis-aligned mixtures of gaussians with no separation assumption. In Gábor Lugosi and Hans Ulrich Simon, editors, *Learning Theory*, pages 20–34, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg. 15

[GHS12] Craig Gentry, Shai Halevi, and Nigel P. Smart. Fully homomorphic encryption with polylog overhead. In David Pointcheval and Thomas Johansson, editors, *Advances in Cryptology – EUROCRYPT 2012*, volume 7237 of *Lecture Notes in Computer Science*, pages 465–482, Cambridge, UK, April 15–19, 2012. Springer Berlin Heidelberg, Germany. 7

[GL89] O. Goldreich and L. A. Levin. A hard-core predicate for all one-way functions. In *Proceedings of the Twenty-First Annual ACM Symposium on Theory of Computing*, STOC '89, page 25–32, New York, NY, USA, 1989. Association for Computing Machinery. 37, 38

[GO96] Oded Goldreich and Rafail Ostrovsky. Software protection and simulation on oblivious rams. *J. ACM*, 43(3):431–473, May 1996. 6

[Gol01] Oded Goldreich. *Foundations of Cryptography: Volume 1, Basic Tools.* Cambridge University Press, Cambridge, UK, 2001. 13

[HHC$^+$23] Alexandra Henzinger, Matthew M. Hong, Henry Corrigan-Gibbs, Sarah Meiklejohn, and Vinod Vaikuntanathan. One server for the price of two: Simple and fast single-server private information retrieval. In Joseph A. Calandrino and Carmela Troncoso, editors, *32nd USENIX Security Symposium, USENIX Security 2023, Anaheim, CA, USA, August 9-11, 2023*, pages 3889–3905. USENIX Association, 2023. 5, 6

[HJK$^+$16] Dennis Hofheinz, Tibor Jager, Dakshita Khurana, Amit Sahai, Brent Waters, and Mark Zhandry. How to generate and use universal samplers. In Jung Hee Cheon and Tsuyoshi Takagi, editors, *Advances in Cryptology – ASIACRYPT 2016, Part II*, volume 10032 of *Lecture Notes in Computer Science*, pages 715–744, Hanoi, Vietnam, December 4–8, 2016. Springer Berlin Heidelberg, Germany. 27

[HR24] Justin Holmgren and Ron Rothblum. Linear-size boolean circuits for multiselection. In Rahul Santhanam, editor, *39th Computational Complexity Conference, CCC 2024, July 22-25, 2024, Ann Arbor, MI, USA*, volume 300 of *LIPIcs*, pages 11:1–11:20. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2024. 5

[IKO05] Yuval Ishai, Eyal Kushilevitz, and Rafail Ostrovsky. Sufficient conditions for collision-resistant hashing. In Joe Kilian, editor, *TCC 2005: 2nd Theory of Cryptography Conference*, volume 3378 of *Lecture Notes in Computer Science*, pages 445–456, Cambridge, MA, USA, February 10–12, 2005. Springer Berlin Heidelberg, Germany. 2

[IKOS08] Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, and Amit Sahai. Cryptography with constant computational overhead. In Richard E. Ladner and Cynthia Dwork, editors, *40th Annual ACM Symposium on Theory of Computing*, pages 433–442, Victoria, BC, Canada, May 17–20, 2008. ACM Press. 2, 7

[IP07] Yuval Ishai and Anat Paskin. Evaluating branching programs on encrypted data. In Salil P. Vadhan, editor, *Theory of Cryptography, 4th Theory of Cryptography Conference, TCC 2007, Amsterdam, The Netherlands, February 21-24, 2007, Proceedings*, volume 4392 of *Lecture Notes in Computer Science*, pages 575–594. Springer, 2007. 8, 9

[ISW24] Yuval Ishai, Elaine Shi, and Daniel Wichs. PIR with client-side preprocessing: Information-theoretic constructions and lower bounds. In Leonid Reyzin and Douglas Stebila, editors, *Advances in Cryptology – CRYPTO 2024, Part IX*, volume 14928 of *Lecture Notes in Computer Science*, pages 148–182, Santa Barbara, CA, USA, August 18–22, 2024. Springer, Cham, Switzerland. 6

[KLP68] T. Kasami, Shu Lin, and W. Peterson. New generalizations of the reed-muller codes–i: Primitive codes. *IEEE Transactions on Information Theory*, 14(2):189–199, 1968. 18

[KO97] Eyal Kushilevitz and Rafail Ostrovsky. Replication is NOT needed: SINGLE database, computationally-private information retrieval. In *38th Annual Symposium on Foundations of Computer Science*, pages 364–373, Miami Beach, Florida, October 19–22, 1997. IEEE Computer Society Press. 1, 5, 8, 11

[LMW23] Wei-Kai Lin, Ethan Mook, and Daniel Wichs. Doubly efficient private information retrieval and fully homomorphic ram computation from ring lwe. In *Proceedings of the 55th Annual ACM Symposium on Theory of Computing*, STOC 2023, page 595–608, New York, NY, USA, 2023. Association for Computing Machinery. 1, 2, 5, 6, 7

[LMW25] Wei-Kai Lin, Ethan Mook, and Daniel Wichs. Black box crypto is useless for doubly efficient PIR. Cryptology ePrint Archive, Paper 2025/552, 2025. 7

[LPR10] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. On ideal lattices and learning with errors over rings. In Henri Gilbert, editor, *Advances in Cryptology - EUROCRYPT 2010, 29th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Monaco / French Riviera, May 30 - June 3, 2010. Proceedings*, volume 6110 of *Lecture Notes in Computer Science*, pages 1–23. Springer, 2010. 1

[LWYY24] Hanlin Liu, Xiao Wang, Kang Yang, and Yu Yu. The hardness of LPN over any integer ring and field for PCG applications. In Marc Joye and Gregor Leander, editors, *Advances in Cryptology - EUROCRYPT 2024 - 43rd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zurich, Switzerland, May 26-30, 2024, Proceedings, Part VI*, volume 14656 of *Lecture Notes in Computer Science*, pages 149–179. Springer, 2024. 4

[OS07] Rafail Ostrovsky and William E. Skeith, III. A survey of single-database private information retrieval: Techniques and applications (invited talk). In Tatsuaki Okamoto and Xiaoyun Wang, editors, *PKC 2007: 10th International Conference on Theory and Practice of Public Key Cryptography*, volume 4450 of *Lecture Notes in Computer Science*, pages 393–411, Beijing, China, April 16–20, 2007. Springer Berlin Heidelberg, Germany. 11

[RR22] Noga Ron-Zewi and Ron D. Rothblum. Proving as fast as computing: succinct arguments with constant prover overhead. In Stefano Leonardi and Anupam Gupta, editors, *54th Annual ACM Symposium on Theory of Computing*, pages 1353–1363, Rome, Italy, June 20–24, 2022. ACM Press. 7

[RSS14] Yuval Rabani, Leonard J. Schulman, and Chaitanya Swamy. Learning mixtures of arbitrary distributions over large discrete domains. In *Proceedings of the 5th Conference on Innovations in Theoretical Computer Science*, ITCS '14, page 207–224, New York, NY, USA, 2014. Association for Computing Machinery. 15

[SWP00] Dawn Xiaodong Song, David Wagner, and Adrian Perrig. Practical techniques for searches on encrypted data. In *Proceedings of the 2000 IEEE Symposium on Security and Privacy*, pages 44–55. IEEE, 2000. 6

[Vid03] Rene Esteban Vidal. *Generalized principal component analysis (gpca): an algebraic geometric approach to subspace clustering and motion segmentation.* PhD thesis, 2003. AAI3121739. 15

[VZ25] Vinod Vaikuntanathan and Or Zamir. Improving algorithmic efficiency using cryptography. *CoRR*, abs/2502.13065, 2025. 6, 11, 30

[YZW$^+$19] Yu Yu, Jiang Zhang, Jian Weng, Chun Guo, and Xiangxue Li. Collision resistant hashing from sub-exponential learning parity with noise. In *Advances in Cryptology – ASIACRYPT 2019: 25th International Conference on the Theory and Application of Cryptology and Information Security, Kobe, Japan, December 8–12, 2019, Proceedings, Part II*, page 3–24, Berlin, Heidelberg, 2019. Springer-Verlag. 3, 15

[ZLTS23] Mingxun Zhou, Wei-Kai Lin, Yiannis Tselekounis, and Elaine Shi. Optimal single-server private information retrieval. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 395–425. Springer, 2023. 6

A Proof of Lemma 4.2

Lemma 4.2 (From LPN to Matrix-LPN)**.** *(k, ε)-LPN implies (k, n, ε)-$\mathsf{Matrix\text{-}LPN}$, for any field $\mathbb{F}$, polynomials $n(\lambda) > k(\lambda) \geq \lambda$, and noise rate $\varepsilon(\lambda)$.*

Proof. We define n hybrid distributions as follows. For $j = 0, \ldots, n$, the distribution D^j consists of $(\mathbf{a}_i, b_{i,1}, \ldots, b_{i,n})$ for $i \in [m]$ such that: for $j' \leq j$, $b_{i,j'} = \mathbf{a}_i^\intercal \mathbf{s}_{j'} + r_{i,j'} \cdot u_{i,j'}$ where $\mathbf{s}_{j'}$ is the j'-th row of a uniformly random matrix $S \leftarrow \mathbb{F}^{k \times n}$, $\mathbf{a}_i \leftarrow \mathbb{F}^k$, $r_{i,j'} \leftarrow \mathsf{Ber}(\varepsilon)$ and $u_{i,j'} \leftarrow \mathbb{F} \setminus \{0\}$, and, for $j' > j$, $b_{i,j'} \leftarrow \mathbb{F}$. If an adversary $\mathcal{A}$ is able to distinguish between a matrix-LPN samples $(\mathbf{a}_i, \mathbf{a}_i^\intercal \mathbf{S} + \mathbf{r}_i)_{i \in [m]}$ and a uniformly random samples $(\mathbf{a}_i, \mathbf{b}_i)_{i \in [m]}$ then he is able to distinguish between D^j and D^{j-1} for some j. We use such an adversary to break LPN given m samples via the following reduction. For every LPN sample $(\mathbf{a}_i, b_i)$ it takes as input, the reduction generates $j - 1$ bits $b_{i,1}, \ldots, b_{i,j-1}$ by $b_{i,j'} = \mathbf{a}^\intercal \mathbf{s}_{i,j'} + r_{i,j'} \cdot u_{i,j'}$, where $\mathbf{s}_{j'} \leftarrow \mathbb{F}^k$, $r_{i,j'} \leftarrow \mathsf{Ber}(\varepsilon)$ and $u_{i,j'} \leftarrow \mathbb{F}$, and samples uniform $b_{i,j+1}, \ldots, b_{i,n} \leftarrow \mathbb{F}$. The reduction adds $(\mathbf{a}_i, b_{i,1}, \ldots, b_{i,j-1}, b_i, b_{i,j+1}, \ldots, b_{i,n})$ to the input of $\mathcal{A}$.

The reduction runs $\mathcal{A}$ with the input consisting of the m samples produced as above. When $(\mathbf{a}_i, b_i)$ is a uniform sample, then the corresponding sample given to the adversary is from the distribution D^j and, if it is an LPN sample, namely $b_i = \mathbf{a}_i^\intercal \mathbf{s} + r_i \cdot u_i$, then the sample given to $\mathcal{A}$ follows the distribution D^{j-1}. □

B Proof of Lemma 4.1

Lemma 4.1 (From Syndrome Decoding to LPN)**.** *Let $k := k(\lambda) > \lambda$ and $m := m(\lambda)$ be polynomials such that $m > (1 + \delta)k$ for some constant $\delta > 0$, and let $\varepsilon(\lambda) > 0$. Assume that for every non-uniform* $\mathrm{poly}(\lambda)$*-time algorithm $\mathcal{A}$, there is a noticeable δ such that for every $\lambda \in \mathbb{N}$*

$$\Pr[\mathcal{A}(\mathbf{H}, \mathbf{y}) = \mathbf{e}] \leq 1 - \delta(\lambda),$$

where $\mathbf{H} \leftarrow \mathbb{F}^{(m-k) \times m}$ and $\mathbf{y} = \mathbf{He}$ for $\mathbf{e} \in \mathbb{F}^m$ sampled from $\mathsf{Ber}^m(\varepsilon)$ *conditioned on its Hamming weight is $\varepsilon \cdot m$. Then, the (k, ε, m)-*LPN *assumption holds.*

Proof. Assume towards contradiction that LPN with the specified parameters does not hold. Then, there exists a distinguisher $\mathcal{D}$ that, on inputs $\mathbf{A} \leftarrow \mathbb{F}^{k \times m}$ and $\mathbf{v} \in \mathbb{F}^m$ can distinguish between the case where $\mathbf{v} = \mathbf{As} + \mathbf{e}$ for $\mathbf{s} \leftarrow \mathbb{F}^k$ and $\mathbf{e} \leftarrow \mathsf{Ber}^m(\varepsilon)$, and the case where $\mathbf{v}$ is uniform.

First, we convert the LPN distinguisher to an algorithm $\mathcal{A}$ that solves the search version of LPN with probability all but negligible. In search-LPN, $\mathcal{A}$ is given an LPN sample $(\mathbf{A}, \mathbf{As} + \mathbf{e})$ and its goal is to output the secret $\mathbf{s} \in \mathbb{F}^k$. To that end, we recall the search-to-decision reduction from [AIK07] which is based on the Goldreich-Levin theorem for predicting hardcore bits [GL89]. First, let us demonstrate how on input $(\mathbf{A}, \mathbf{v} = \mathbf{As} + \mathbf{e})$, we can use $\mathcal{D}$ to compute a Goldreich-Levin hardcore bit $\mathbf{r}^\intercal\mathbf{s}$, for a uniformly random $\mathbf{r} \leftarrow \mathbb{F}^k$: (1) Sample uniformly random $\mathbf{z} \leftarrow \mathbb{F}^m$. (2) Send $(\mathbf{A}', \mathbf{v})$ to $\mathcal{D}$, where $\mathbf{A}' = \mathbf{A} - \mathbf{z}\mathbf{r}^\intercal$, and return its output.

The hardcore-bit is computed correctly with probability $1/2 + \text{poly}(\lambda)$ since it holds that $\mathbf{v} = \mathbf{A}'\mathbf{s} + \mathbf{z}\mathbf{r}^\intercal\mathbf{s} + \mathbf{e}$. Therefore, if $\mathbf{r}^\intercal\mathbf{s} = 0$, then $\mathbf{v} = \mathbf{A}'\mathbf{s} + \mathbf{e}$ and, otherwise $\mathbf{v}$ is uniformly random. Since $\mathcal{D}$ can distinguish between these two cases with advantage $1/\text{poly}(\lambda)$ by assumption, we guess the hardcore predicate with the same advantage. Due the Goldreich-Levin theorem [GL89], such a guesser can be turned into an extractor of $\mathbf{s}$ that succeeds to compute $\mathbf{s}$ in polynomial time with inverse-polynomial probability *over the choice of his randomness.* Hence, by repeating the extractor sufficiently many times with i.i.d. random coins, we obtain a solver $\mathcal{A}$ for search-LPN that reaches success probability $1 - \delta(\lambda)$, for a negligible $\delta(\lambda)$.

Next, we convert the LPN solver $\mathcal{A}$ to an algorithm $\mathcal{A}'$ that solves "dual-LPN", or syndrome decoding, where, given $(\mathbf{H}, \mathbf{He})$, for uniformly random $\mathbf{H} \leftarrow \mathbb{F}^{(m-k)\times m}$ and $\mathbf{e} \leftarrow \mathsf{Ber}^m(\varepsilon)$, the task is to compute $\mathbf{e}$. Given $(\mathbf{H}, \mathbf{w})$, our dual-LPN solver $\mathcal{A}'$ samples a uniformly random $\mathbf{A} \in \mathbb{F}^{m\times k}$ such that $\mathbf{HA} = \mathbf{0}$ (note that a uniformly random $\mathbf{H}$ of these dimensions is full-rank with probability all but negligible since $k > \lambda$) and a uniformly random $\mathbf{v} \in \mathbb{F}^m$ satisfying $\mathbf{Hv} = \mathbf{w}$. He then sends $(\mathbf{A}, \mathbf{v})$ to the LPN solver and gets back $\mathbf{s}$. Given $\mathbf{s}$, $\mathcal{A}'$ outputs $\mathbf{e} = \mathbf{v} - \mathbf{As}$.

When $\mathbf{H}$ is uniform, then so is $\mathbf{A}$. Given $\mathbf{w} = \mathbf{He}$, $\mathbf{v}$ is sampled from the space of all vectors in the coset $\ker(\mathbf{H}) + \mathbf{e}$. Since $\mathbf{A}$ is full rank with overwhelming probability (since $m > (1 + \delta)k$), the kernel is spanned by $\mathbf{A}$ and the coset from which $\mathbf{v}$ is sampled uniformly is precisely the set $\{\mathbf{As} + \mathbf{e} \mid \mathbf{s} \in \mathbb{F}^k\}$. Hence, $\mathcal{A}'$ returns the correct $\mathbf{e}$ with probability all but negligible.

The last remaining step is to convert the instance $(\mathbf{H}, \mathbf{He})$, where $\mathbf{e} \leftarrow \mathsf{Ber}^m(\varepsilon)$, to an instance where $\mathbf{e}$ is sampled conditioned on its Hamming weight is exactly $\varepsilon \cdot m$. For that, we note that the probability for $\mathbf{e} \leftarrow \mathsf{Ber}^m(\varepsilon)$ to have exactly its average weight is $\binom{m}{\varepsilon m}\varepsilon^{\varepsilon m}(1-\varepsilon)^{(1-\varepsilon)m} \geq 1/\varepsilon m$, which is at least inverse-polynomial. Hence, if $\mathcal{A}'$ finds $\mathbf{e}$ with probability all but negligible when $\mathbf{e}$ is sampled from the unconditional distribution of Bernoullis, then it does so when $\mathbf{e}$ is sampled conditionally as above. $\square$