Università Commerciale "Luigi Bocconi"
PhD School

PhD program in Statistics and Computer Science

Cycle: XXXVI

Disciplinary Field (code): INF/01

Fast Protocols for Secret-Key Private Information Retrieval

Advisor: Alon Rosen

Co-Advisor: Yuval Ishai

PhD Thesis by

Caicai Chen

ID number: 3147178

Year: 2026

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 $\mathrm{polylog}(N)$ bits of $\widehat{x}$ per query. This requires $|\widehat{x}| \geq N \cdot \mathrm{polylog}(N)$, and even larger server circuit size.

In Chapter 2, 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 $\varepsilon > 0$, under 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 conjectures 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.

In Chapter 3, observing that the fine-grained security setting, where the adversary's runtime is bounded by a fixed polynomial in the input size, enables meaningful efficiency gains in our context, we design and implement the first practical doubly efficient secret-key PIR scheme based on the protocol proposed by Boyle *et al.* and Canetti *et al.* (TCC 2017). In addition, we propose several optimizations that significantly improve the cost of the original approach. Setting for relaxed security, our implementation offers a stateless client, a lightweight online phase, near optimal download rate, and scalability to large databases, outperforming recent works on practical PIR with preprocessing.

Acknowledgements

This thesis would not have been possible without the guidance, support, and encouragement of many people along the way. I am deeply grateful to each of them for making this journey both fruitful and enjoyable.

First and foremost, I owe my deepest thanks to my advisor, Prof. Alon Rosen, who was always incredibly welcoming and generous with his time. He encouraged me to explore the topics that truly interested me, to reach out and collaborate with others, and to approach each new idea with confidence. Whenever I hit a roadblock, he was there with just the right encouragement and insight to help me push through. No matter when I needed help, he was always available and ready to support every idea I brought to him.

I consider myself exceptionally fortunate to have Prof. Yuval Ishai as my advisor. He brings a wealth of original ideas and a remarkably deep perspective to every topic we discuss. No matter when I reach out, he consistently replies with thoughtful, lightning-fast answers, supplies me with abundant references, and offers crystal-clear guidance. Beyond the technical insights, he has taught me how to structure my thinking, approach problems methodically, and tackle research challenges with confidence and rigor.

I was deeply fortunate to have Prof. Luca Trevisan as my very first Ph.D. advisor. Tragically, he passed away last year, but his early mentorship laid the groundwork for my entire academic journey. With no prior research experience, I relied on his unfailing patience as he taught me how to frame a research question, how to break complex problems into manageable steps, and how to explore the literature effectively. I was devastated by his loss, yet his guidance and the confidence he instilled in me remain a constant source of inspiration.

My heartfelt thanks go to the members the entire Cryptography and Theory of Computer Science groups at Bocconi University. In particular, I'm deeply grateful to my

co-authors Tamer Mour and Chris Johns: our countless discussions about papers, ideas, and research challenges have been an indispensable part of both my academic progress and personal growth.

I would also like to thank Prof. Katerina Sotiraki for graciously hosting me at Yale University. Her invitation opened the door to the Yale Crypto group's vibrant community, where I had the pleasure of meeting many new colleagues and friends whose insights and camaraderie enriched my research experience.

I have been fortunate to be surrounded by lifelong friends—Guoman She, Qiaoni Shi, Linghuan Wang, Shengyu Zhao, Wanli Zhao, and Kai Zhu—whose companionship exists entirely outside of academia. Whether we're gathered around a board game, experimenting with new recipes in the kitchen, or setting off on our next trip together, their friendship has filled this journey with laughter, adventure, and balance. These shared moments remind me of life's richness beyond the research and have been a vital source of joy and perspective.

Finally, I extend my deepest gratitude to my family. I am forever indebted to my parents, Shaohua Lu and Kui Chen, for the love, opportunities, and experiences that shaped who I am today. To my in–laws, Mingjuan Niu and Shaoping Feng, thank you for your unwavering support and encouragement. And to my best friend, greatest companion, and the love of my life, Tangren Feng, your patience and understanding have carried me through every challenge. I cherish our family moments with Stefanie and Elio—my daughter and son—whose joy and curiosity remind me daily of what truly matters.

Thank you all.

Contents

Acknowledgements **ii**

1 Secret-Key Private Information Retrieval **1**

1.1 Introduction 1

1.1.1 PIR with preprocessing 1

1.1.2 Secret-Key PIR 2

1.1.3 Comparison with Related Models 3

2 Secret-Key PIR from Random Linear Codes **4**

2.1 Introduction 4

2.1.1 Our Results 5

2.1.2 Secret-Key PIR from Learning Parity with Noise 6

2.1.3 Better Efficiency via Learning Subspace with Noise 7

2.1.4 Minimizing Server Complexity via Splitting 9

2.1.5 Minimizing Client Complexity 13

2.1.6 Related Work 13

2.2 Technical Overview 16

2.2.1 Balancing Communication 18

2.2.2 Further Reducing Communication via Folding 18

2.2.3 Analysis of Basic LSN 20

2.2.4 Replacing LSN by LPN 21

2.2.5 Minimizing Server Circuit Complexity 22

2.2.6 Minimizing Client Computation 22

2.3 Preliminaries 25

2.3.1 Standard Cryptographic Notions 26

2.3.2 Secret-key Private Information Retrieval 27
2.3.3 Learning Parity with Noise 29
2.4 Learning Subspace with Noise 30
2.4.1 The Basic LSN Assumption 31
2.4.2 Split-LSN 36
2.4.3 A Unified LSN Framework 37
2.4.4 LPN Implies Sparse-Noise LSN 39
2.5 Base Protocol 43
2.6 Improving Communication via Folding 49
2.7 Improving Client Runtime 55
2.7.1 Cubic-Root Client Runtime under LPN 56
2.7.2 Composing with Standard PIR 59

3 Fast Secret-Key PIR in the Fine-Grained Setting 60
3.1 Introduction 60
3.1.1 Our Contribution 60
3.1.2 Detailed Overview 63
3.1.3 Our Techniques 71
3.2 Preliminaries 75
3.2.1 Reed-Muller Codes 75
3.3 Background: PIR via Reed-Muller 77
3.3.1 Complexity 78
3.3.2 Security 79
3.4 Our Base Scheme 81
3.4.1 The Protocol 83
3.4.2 Cryptanalysis 85
3.5 Implementation 90

3.5.1 The Pseudo-Random Permutation 91
3.5.2 AES for Data Encryption 93
3.6 Performance 93

Bibliography **113**

Appendix A: Proofs for Chapter 2 **113**
.1 Proof of Lemma 2.4.2 113
.2 Proof of Lemma 2.4.1 113

Appendix B: Further Cryptanalysis for Chapter 3 **116**
.3 Preliminary Statistical Analysis 116
.4 A Birthday Attack 119
.5 A Statistical Attack on The Degree-1 Case 119
.6 Resilience against 1-Sample Statistics 121
.7 Resilience against Intersection-Size Statistics 122
.8 Sidelnikov-Shestakov Attack 129

Chapter 1

Secret-Key Private Information Retrieval

1.1 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 setting.

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.

1.1.1 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.

1.1.2 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 [BHW19a, BW21, BHMW21], this assumption is still quite poorly understood.

1.1.3 Comparison with Related Models

An alternative setting for PIR with preprocessing allows any client to maintain a database-dependent state, or "hint," which is computed during a preprocessing stage and can be used to access the ("sever-owned") database. Recent works have shown how to leverage such hints towards concretely efficient PIR schemes with sublinear server computation [CK20, KCG21, ZPSZ23, MIR23, LP24]. The main disadvantage of these schemes is that the client needs to download, store, and update a database-dependent state that may be quite large. A secondary limitation is that the server's computation, and typically also communication, must scale (at least) linearly with the square-root of the database size.

Another model that resembles sk-PIR is that of *oblivious RAM* (ORAM) [Gol87, GO96a]. In ORAM, a client delegates RAM storage to a server and is able to perform reads and writes privately. The main disadvantage of ORAM compared to sk-PIR is that both client and server must maintain a state in-between the different queries. In contrast, sk-PIR supports not only a stateless client, but also a stateless server. This is particularly important for use cases where the same (distributed) secret key is used for many concurrent reads. For instance, in a distributed-client setting, multiple MPC protocols can access the same copy of the encoded database concurrently. Alternatively, the same secret key may be concurrently used by multiple devices/threads owned by the same user, with only a single global encoded database.[1] Additionally, our sk-PIR scheme achieves better than $\sqrt{n}$ communication while still requiring minimal interaction of just one round of questions and answers. ORAM inherently has a higher round complexity [CDH20].

[1] While there are parallel ORAM solutions, they are more complicated and need to deal with intricate state synchronization issues that are entirely avoided by sk-PIR.

Chapter 2

Secret-Key PIR from Random Linear Codes[1]

2.1 Introduction

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 $N \cdot \mathrm{polylog}(N)$. 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] This chapter is based on joint work with Yuval Ishai, Tamer Mour and Alon Rosen [CIMR25].

2.1.1 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 [BFKL94] 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 variant of the *Learning Subspace with Noise* (LSN) conjecture [DKL09] on learning hidden linear subspaces of $\mathbb{F}_2^n$ with noise.

The class of "Learning Subspace with Noise" assumptions, first proposed by Dodis, Kalai and Lovett in the context of leakage-resilient cryptography [DKL09], considers the pseudorandomness of noisy samples from a secret linear code. This and related LSN-style 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 relies on the conjectured

hardness of what we call the *split-LSN* problem, a more specialized variant of the original LSN assumption from [DKL09]. In spite of its more specialized nature, split-LSN may offer a better level of security than the original variant of LSN, potentially resisting sub-exponential time algebraic attacks that break the original LSN in the constant-rate regime (see Section 2.1.3).

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.

2.1.2 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 2.3.4).

Theorem 2.1.1 (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 2.1.1 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.[2]

2.1.3 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 that contains it. We start with the original version of LSN that uses a simple mixture noise.

Definition 2.1.1 (Basic LSN [DKL09])**.** *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*

[2] 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.

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 LSN assumption over $\mathbb{F}_2$ is equivalent to the standard LPN assumption with noise rate $\mu/2$ [DKL09]. We obtain several other useful results on LSN and its relation with LPN.

Theorem 2.1.2 (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.*
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 3.1.2 for proof overview. The distinguisher from part (1) will guide our parameter choice for LSN. A similar distinguisher is attributed in [DKL09] to Ran Raz, but is only described in an unpublished full version. We provide here a proof for completeness, which also covers general fields $\mathbb{F}$. A similar result to part (2), but with the weaker conclusion of LPN with noise rate close to $1/2$, is implied by [DKL09, Theorem 1] (which considers a stronger notion of security with auxiliary input). Part (3) serves as a basis for the LPN-based construction of Theorem 2.1.1, which follows as a special case of an sk-PIR construction based on LSN with general noise.

We make the following conjecture, which is qualitatively similar to (but slightly stronger than) the "LSN assumption" put forward in [DKL09].

Conjecture 2.1.1 (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 our LSN conjecture, we can make the storage nearly optimal, or alternatively get a weak notion of "doubly efficient" PIR.

Theorem 2.1.3 (Low-storage or sublinear-computation sk-PIR from LSN, Informal)**.** *Suppose Conjecture 2.1.1 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.

2.1.4 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[3] $\mathbf{c}_i + V_i$, 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.

[3]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.

Definition 2.1.2 (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 [AG11a, LWYY24] and references therein. However, such a regular variant of LSN would require r to be super-constant, 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 [BHW19a, BW21, BHMW21]. In this protocol, the secret code C is an RM code (the dual of an RM code used for encoding the database), 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 2.1.2 (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 algebraic attacks seem less effective. In particular, while part (1) of Theorem 2.1.2 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 2.4.2 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 2.1.4 (Server-efficient sk-PIR from split-LSN, Informal)**.** *Suppose Conjecture 2.1.2 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 $N \cdot \mathrm{polylog}(N)$. Number-theoretic PIR schemes (e.g., [KO97]) have an even higher computational cost.

Under a variant of Conjecture 2.1.2 with rate $\rho = o(1)$, where hardness of split-LSN is assumed to hold against *quasi-polynomial* time algorithms, the result from Theorem 2.1.4 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 2.1.3, and has the advantage of low storage overhead. 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 crude sense of potential practical utility: in the above sk-PIR protocol, the amortized server work for processing 64 bits of the database requires two XOR and two AND operations of 64-bit strings. In fact, since each matrix-vector product over $\mathbb{F}_2$ requires computing the XOR of only *half* the matrix columns on average, a RAM implementation will involve only *one* 64-bit XOR per 64 bits of the database, excluding the cost of memory access. For comparison, a typical instantiation of the SimplePIR protocol [HHC+23] (which uses *public* preprocessing) requires 8 multiplications and 8 additions over $\mathbb{Z}_{2^{32}}$ per 64 database bits, for a database of 2^{33} bits. 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 hard-

ware, 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.

2.1.5 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.

2.1.6 Related Work

Learning subspace with noise. The basic LSN problem (Definition 2.1.1) was first introduced in [DKL09], where it was used to obtain applications in leakage-resilient cryptography. Similar applications were later derived from the standard LPN assumption [YZ21].

The search version of LSN was studied in [CDV21] in the broader context of learning mixtures of uniform distributions over linear subspaces of $\mathbb{F}_2^n$. The basic LSN experiment involves a mixture of a uniform distribution over a secret linear subspace $C \subset \mathbb{F}_2^n$ with a

uniform distribution over the entire space $\mathbb{F}_2^n$. While efficient search-to-decision reductions for LPN are known [BFKL94, AIK09], the existence of such a reduction for any flavor of LSN is still open.

In the context of PIR, structured variants of LSN are implicit in the doubly-efficient sk-PIR constructions from [BIPW17, CHR17], which were further analyzed in [BHW19a, BW21, BHMW21]. In addition to using a highly structured class of codes (permuted Reed-Muller codes) and different kinds of noise, another difference from LSN is that the (shifted) codewords in these constructions are picked subject to having low Hamming weight, rather than uniformly from a secret linear code.

Alternative PIR models. There is a large body of works studying alternative models for PIR with preprocessing, including Oblivious RAM (ORAM) [GO96b] and other PIR models [CK20, ZLTS23]. In the above 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.

Follow-up work. Benhamouda et al. [BCH+25] apply the LSN-based approach of the current work to the problem of *encrypted matrix-vector product* (EMVP). Similarly to sk-PIR, EMVP enables a client to make an unbounded number of queries to a big encrypted

input. In the EMVP case, the input is a matrix $\mathbf{M} \in \mathbb{F}^{m \times \ell}$ (where $\mathbb{F}$ is typically a large finite field), each query is a vector $\mathbf{q}_i \in \mathbb{F}^{\ell}$, and the output is the matrix-vector product $\mathbf{M} \cdot \mathbf{q}_i$, which can be further post-processed by another secure computation protocol. EMVP is motivated by several cryptographic use cases, including secure fuzzy search. In addition to proposing a new 1-dimensional variant of split-LSN that minimizes the costs of EMVP over big fields, the results of [BCH+25] include analysis and optimizations that are relevant also to the case of sk-PIR. The latter include the use of structured codes for minimizing the complexity of EMVP clients, concretely efficient constructions of "trapdoored matrices" (a primitive we use for improving the client's computation – see 2.2.6), protocols for distributing EMVP clients, and suggested parameters for the underlying (split-)LSN assumptions based on a concrete analysis of relevant classes of algebraic attacks.

2.2 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

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 2.1: Noisy sk-PIR protocol with short answers.

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. 2.1. 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.

2.2.1 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].[4]

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. 2.1, 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 with $\approx \sqrt{N}$ bits of communication.

2.2.2 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

[4]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 satisfy this property.

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 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.

This generic blueprint does not always apply in our secret-key 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. These responses do not necessarily conform to the structure of an encoded database and it is not clear how to make them so without first communicating them to the client.

To get around this difficulty, we rely on the *linearity* of the database encoding. We observe that if we carry out the above outline with our base protocol, it is possible to encode the original database columns in a way that results in 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 us to apply another PIR query over the resulting column. The idea can be carried out recursively to further reduce the communication. By folding the database any constant number of times, we obtain communication of $O(N^{\varepsilon})$ 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 $N^{o(1)}$. If we consider the database size as the security parameter, this requires assuming quasi-polynomial hardness of the underlying LSN problem for security.

Applying linear codes along different dimensions of high-dimensional data can be captured by the notion of a *tensor code*. The folded protocol can be 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.

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).

2.2.3 Analysis of Basic LSN

We now outline the proofs of the three LSN-related facts stated in Theorem 2.1.2.

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. (A similar result is proved in an unpublished full version of [DKL09], where it is attributed to Ran Raz.) The distinguisher can be seen as 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 more often than 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 2.4.2 for details.

Basic 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 2.4.1 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 2.4.3 for more details.

2.2.4 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. One can naturally trade the strength of the LPN assumption for storage. For example, with $n = k + k^{9/10}$ and $\eta > 9/10$ (a lower LPN noise rate), the storage is $O(N^{10/9})$.

2.2.5 Minimizing Server Circuit Complexity

Our split-LSN Conjecture (Conjecture 2.1.2) 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 $\approx 4N$.

2.2.6 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. 2.1). (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, Ste98, CMS99, Lip05, 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. 2.1, 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 2.1.2), 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 $O(N^{1/3+\varepsilon})$.

2.3 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.

2.3.1 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 2.3.1 (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* poly(λ)*-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 2.3.2 (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.

2.3.2 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 2.3.3 (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.

2.3.3 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 2.3.4 (LPN)**.** *The* (decisional) learning parity with noise *assumption* (k, ε)-LPN

over $\mathbb{F}$ *asserts that for any polynomial* $m := m(\lambda)$ *we have:*

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

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.

2.4 Learning Subspace with Noise

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

2.4.1 The Basic LSN Assumption

We start by discussing the basic LSN variant from [DKL09] (Definition 2.1.1). 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 2.4.1 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. This

can be compared to [DKL09, Theorem 1], which shows that LSN implies a stronger leakage-resilient flavor of LPN, but with a high noise rate (close to 1/2). In our implication the LPN noise rate vanishes.

- In Section 2.4.1 we present the "tensored rank" distinguisher for LSN, which is attributed in [DKL09] to Ran Raz and can be seen as a simplified version of the algorithm for the search problem from [CDV21].

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 $|\mathbb{F}| \leq \text{poly}(\lambda)$ (for which LPN is known to have a search-to-decision reduction).

Proposition 2.4.1 (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 [AIK09].

Lemma 2.4.1 (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* $\text{poly}(\lambda)$*-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 2.4.1 to Appendix .2 and proceed to prove Proposition 2.4.1. 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 2.4.1. 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. □

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 how to leverage a larger number of samples than the LPN-based attack to break the (decision) LSN assumption with any positive constant noise rate μ. In light of this algorithm, we must require $1-\mu$ to be sub-constant in Conjecture 2.1.1.

Proposition 2.4.2 (Rank attack against LSN with low noise rate)**.** *For any field $\mathbb{F}$ of size q, any constant $d > 0.99e \cdot q \cdot \ln^{1/d}(q)$, and any polynomials k, n and noise rate $\mu \leq 1 - (k^d + 1)/n^d$, there exists a distinguisher that breaks (k, n, μ)-LSN in time $O(n^d)$ using n^d samples.*

Proof. The distinguisher is given n^d samples $\mathbf{v}_1, \ldots, \mathbf{v}_{n^d} \in \mathbb{F}^n$ 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 at rate μ. 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 an $n^d \times n^d$ 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 non-zero with probability μ, then, in expectation, there are $(1-\mu)n^d \geq k^d + 1$ noiseless samples where $\mathbf{v}_i = \mathbf{a}_i^\intercal \mathbf{C}$. Since the noise in the samples is i.i.d., the number of noiseless samples is follows the Binomial distribution and it is at least $k^d + 1$ with probability at least $1/(k^d+1)$. Assuming this is the case, there exist at least k^d+1 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 $\mathbf{V}$ is not full with probability $\Omega(1/k^d)$.

On the other hand, we argue that if the samples $\mathbf{v}_i$ are uniform, then the rank of $\mathbf{V}$ is full with overwhelming probability. 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 $\mathbf{v}_1, \ldots, \mathbf{v}_{n^d}$. 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 n^d 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 n^d uniform assignments are all roots of a given polynomial is at most $(1-1/q^d)^{n^d} = e^{-(n/q)^d}$. The number of all such polynomials is bounded by $q^{\binom{n+d}{d}} \leq q^{(e(n+d)/d)^d} = \exp(\ln q \cdot e^d \cdot (1+n/d)^d) < \exp(0.99^d(d+n)^d/q^d) < \exp((\alpha n)^d/q^d)$ for some constant $\alpha < 1$. Hence, the probability that there exists a polynomial that vanishes on all $\mathbf{v}_i$ is at most $e^{-\Omega(n^d)}$ by union bound. □

2.4.2 Split-LSN

Our most efficient sk-PIR constructions rely on the split-LSN conjecture (Conjecture 2.1.2), 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 2.1.4), 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 2.4.2 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 [AG11a] 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 $\mathbf{v}^{\leq d}$ 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 $\leq d$ 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 $\mathbf{v}_i^{\leq d}$ 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$\leq d$ 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.

2.4.3 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 basic LSN (Definition 2.1.1) split-LSN (Definition 2.1.2) and a new notion of sparse-noise LSN (see Definition 2.4.3 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 2.4.1 (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 2.1.1 and 2.1.2 as well as a new sparse-noise LSN variant as instances of general LSN.

Definition 2.4.2 (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 basic 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.

2.4.4 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 basic LSN (Definition 2.1.1). 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 2.4.3 (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* $\mathbf{R} \in \mathsf{Rank}_k(\mathbb{F}^{k\times k})$*,* $\mathbf{r}_i \leftarrow \mathsf{Ber}^{n-k}(\varepsilon)$ *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 $\mathbf{R} \leftarrow \mathsf{Rank}_k(\mathbb{F}^{k\times k})$. 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 2.4.3.

(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 2.4.3) is a generalization of the connection made in [CDV21] for the search variants of the problems. We formalize this consequence in Corollary 2.4.1.

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

Proposition 2.4.3 (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 2.4.3, where $n = k + 1$, implies the following.

Corollary 2.4.1 ([DKL09, CDV21])**.** *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 2.4.3, 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 2.4.4 (Matrix LPN)**.** *The* (decisional) (k, n, ε)-Matrix-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 .1.

Lemma 2.4.2 (From LPN to Matrix-LPN)**.** (k, ε)-LPN *implies* (k, n, ε)-Matrix-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 2.4.3) using Matrix-LPN as an intermediate problem.

Proof. Let $t = n - k$. Due to Lemma 2.4.2, it is sufficient to show a reduction from (k, t, ε)-Matrix-LPN. Assuming an adversary $\mathcal{A}$ that breaks (k, n, ε)-snLSN, we build an adversary $\mathcal{B}$ that breaks (k, t, ε)-Matrix-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 $\mathbf{d}_i = (\mathbf{a}_i, \mathbf{b}_i)^\intercal \cdot \Pi$.

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 square 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 2.4.3, 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. □

2.5 Base Protocol

We present our base sk-PIR protocol in Fig. 2.2. The protocol may be instantiated based on any LSN assumption satisfying the template laid in Definition 2.4.1.

Lemma 2.5.1 (Base Protocol Security)**.** *The sk-PIR protocol from Fig. 2.2 is secure under* $(k, n, \mathcal{D})$-LSN.

Proof. Consider a hybrid protocol where the outputs of the PRF are replaced by uniformly

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 $(\mathbf{C}, \mathcal{E}) \leftarrow \mathcal{D}(1^\lambda, 1^k, 1^n)$ for $n = k + N_1$.
     - Let $\mathbf{G} \in \mathbb{F}_2^{N_1 \times n}$ be a generator matrix of $\mathbf{D} = \mathbf{C}^\perp$.
     - 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 $\mathbf{c} \leftarrow \mathbf{C}$ and $\langle E \rangle \leftarrow \mathcal{E}$. Let $\mathbf{Q} = \langle \mathbf{Span}(\mathbf{c} + \mathbf{v}_{i_1} + \mathbf{E}) \rangle \in \mathbb{F}_2^{r \times n}$ 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 2.2: Our Base sk-PIR Protocol under $\mathcal{D}$-LSN.

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 2.3.3) 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 $\mathbf{c}_i + E_i$ are $(k, n, \mathcal{D})$-LSN samples and, therefore, are jointly pseudorandom under the corresponding LSN assumption. Since

# in Thm. 2.5.1	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. 2.1.1)	$N^{\frac{1}{2}+\varepsilon}$	$(1+\varepsilon)N$	-	-
		$N^{\frac{1}{2}+\varepsilon}$	$N^{1+\varepsilon}$	-	$\omega(1)\cdot N/\log N$
3	Split-LSN (Conj. 2.1.2)	$N^{\frac{1}{2}+\varepsilon}$	$(1+\varepsilon)N$	$(4+\varepsilon)N$	$(3/4+\varepsilon)N$

Table 2.1: Instantiations of our base protocol from Figure 2.2 under different LPN (Definition 2.3.4) and LSN assumptions. Here, we consider N to be also a security parameter, namely we require that any $\mathrm{poly}(N)$-time adversary has advantage at most $\mathrm{negl}(N)$. See Theorem 2.5.1 for more details.

$\mathbf{Q}_i$ is a function of $(\mathbf{v}_i + \mathbf{c}_i) + E_i$, security under LSN follows by

$$\left(\langle (\mathbf{v}_i + \mathbf{c}_i) + \mathbf{E}_i \rangle \right)_{i\in[m]} \equiv \left(\langle \mathbf{v}_i + (\mathbf{c}_i + \mathbf{E}_i) \rangle \right)_{i\in[m]} \overset{c}{\approx} \left(\langle \mathbf{v}_i + \mathbf{r}_i + \mathbf{E}_i \rangle \right)_{i\in[m]} \equiv \left(\langle \mathbf{r}_i + \mathbf{E}_i \rangle \right)_{i\in[m]} \tag{2.1}$$

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 3.1.2, 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 2.5.1 (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. 2.1.1), 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. 2.1.2), 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 2.5.1 follow the same general recipe of instantiating the protocol from Fig. 2.2.

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 2.5.1.

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 2.4.3 – recall the parameter ε of sparse-noise LSN (see Definition 2.4.3) 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. □

2.6 Improving Communication via Folding

Our folded sk-PIR protocol is presented in Fig. 2.3. It can be seen as a generalization of 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 $t = 1$.)

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

  $$\mathsf{que} = (\mathbf{Q}_1, \ldots, \mathbf{Q}_t) \qquad \mathsf{aux} = ((\mathbf{Q}_1, \ldots, \mathbf{Q}_t), (\mathbf{d}_1, \ldots, \mathbf{d}_t)),$$

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

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 2.6.1 (Folded Protocol Security)**.** *The sk-PIR from Fig. 2.3 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. (2.1). 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 2.2. Communication in the folded protocol can be made an 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 2.6.1 (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* $O(N^{1/(\gamma' t+1)})$.

# in Thm. 2.6.1	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. 2.1.1)	N^{ε}	$(1+\varepsilon)N$	-	-
		N^{ε}	$N^{1+\varepsilon}$	-	$\omega(1) \cdot N/\log^{1/\varepsilon} N$
3	Split-LSN (Conj. 2.1.2)	N^{ε}	$(1+\varepsilon)N$	$(4+\varepsilon)N$	εN
Rem. 2.6.1	Quasi-poly Split-LSN	$N^{o(1)}$	$(1+\varepsilon)N$	$(4+\varepsilon)N$	$o(N)$

Table 2.2: Instantiations of our folded protocol from Figure 2.3 under different LPN (Definition 2.3.4) and LSN assumptions.We consider N to be also a security parameter, namely we require that any $\mathrm{poly}(N)$-time adversary has advantage at most $\mathrm{negl}(N)$. We refer the reader to Theorem 2.6.1 for precise details.

2. *Under the LSN conjecture (Conj. 2.1.1), 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. 2.1.2), 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* $O(N^{1+\varepsilon})$.

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

Remark 2.6.1 (sk-PIR under quasi-polynomial split-LSN)**.** *By instantiating the protocol from Item 3 in Theorem 2.6.1 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 2.1.2) with the same server storage and circuit size, communication $N^{o(1)} + \mathrm{poly}(\lambda)$ *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 2.6.2. *All statements of Theorem 2.6.1 and Remark 2.6.1 hold with respect to public-key PIR with trusted setup in the ideal obfuscation model.*

We proceed to prove Theorem 2.6.1.

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 2.5.1, up to substituting the noise rate with $(1-\mu)^t$.

The alternative protocols with sublinear probe complexity are obtained by the same pre-processing technique that we applied to the base protocol (see the proof of Theorem 2.5.1). 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 2.1.2). 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 $(1 + o(1))4nN_0 < (4 + \varepsilon)N$. □

2.7 Improving Client Runtime

The client in the folded protocols from Theorem 2.6.1 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.

2.7.1 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 2.4.3) 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 2.4.3. 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 2.7.1 (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 2.7.2 (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 2.7.1, 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 2.4.2 (see Appendix .1), 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 2.7.3 (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* $\mathbf{Mv}$ *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 2.7.1. *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 2.6.1 (instance 1) with $t = 2$, using Lemmas 2.7.2 and 2.7.3.

First, by Lemma 2.7.2, we may replace the sparse-noise LSN sampler, which underlies the protocol from Theorem 2.6.1, 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 2.7.2), 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 2.7.3. 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 $O(rn^{1+\varepsilon}) = O(N^{\frac{1}{3}+\varepsilon})$. □

2.7.2 Composing with Standard PIR

Lastly, we use plain-model PIR as a tool to bootstrap any of our small-communication protocols from Theorem 2.6.1 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* 2.6.1).

Proposition 2.7.1. *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 2.6.1 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. 2.2 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. □

Chapter 3

Fast Secret-Key PIR in the Fine-Grained Setting[1]

3.1 Introduction

As established in Chapter 2, secret-key PIR protocols based on random linear codes can achieve strong theoretical guarantees under the Learning Subspace with Noise (LSN) assumption. In this chapter, we revisit the secret-key PIR construction proposed by Boyle *et al.* and Canetti *et al.* [BIPW17, CHR17], that use structured Reed-Muller (RM) codes rather than random linear code. The security of their approach relies on a variant of the LSN assumption that has endured some analysis efforts since then [BHW19b, BHMW21, BW21]. The schemes proposed in these works are impractical. In particular, for a database of size n and security parameter κ, the encoded database size is $\gg \kappa^2 n$. Moreover, the *download rate* is also quite poor, requiring the client to download $\gg \kappa\ell$ bits to retrieve a record of length ℓ.

3.1.1 Our Contribution

We design and implement *Skipper*, the first practical candidate for a secret-key PIR scheme. Our starting point is the observation that the *fine-grained* security setting [Mer78, DVV16, BRSV17], where the adversary's running time is bounded by a fixed polynomial in the input size, may be good enough for PIR, where the input size is very large. The traditional goal of a negligible distinguishing advantage can be similarly relaxed. This gives hope for instantiating the blueprint from [BIPW17, CHR17] with a much better concrete overhead than what the crude asymptotic analysis suggests. We then propose

[1] This chapter is based on joint work with Yuval Ishai, Tamer Mour and Alon Rosen.

several high-level and low-level optimizations that offer significant further improvements to the computation and communication costs.

This results in a framework for practical sk-PIR schemes with the following distinctive features:

- *Stateless client:* The client does not need to store any database-dependent hint, nor should it update a state between different queries. Instead, it only needs to store a short secret key (e.g., a 128-bit AES key) that can be reused for any number of queries on any number of encoded databases.

- *Lightweight online phase:* The online work, including both communication and computation, is competitive with (and typically better than) the best alternative solutions (see Table 3.1). In particular, the online work can beat the $\sqrt{n}$ barrier encountered by other protocols.

- *Optimal download rate:* For databases with record size ℓ, the server-to-client communication is very close to ℓ (and sometimes matches ℓ), even for small values of ℓ.

- *Scalability:* The database encoding is practically feasible even for large databases ($\geq$1GB), increasing their size by 1-2 orders of magnitude.

A comparison of the performance of our schemes to state-of-the-art protocols is given in Table 3.1. As shown in the table, our constructions perform particularly well in terms of online runtime and download rate, significantly improving upon the best existing stateless-client PIR protocols. Some of the instances exhibit a notable improvement in communication as well. Additionally, Table 3.2 signifies the scalability of our solution. Our approach offers these advantages at the expense of a nontrivial blow-up in server storage. Hence, it is particularly useful in applications where such a blow-up is affordable.

The security of Skipper relies on variants of the "permuted low-degree curves" assumption that underlies the original sk-PIR schemes from [BIPW17, CHR17], which are a spe-

cial case of the LSN assumption from Chapter 2 Section 2.4. In a nutshell, the database is encoded using the values of a multivariate polynomial, where each database entry has an associated evaluation point. The polynomial values are encrypted and permuted using the client's secret key. A database entry is retrieved by querying the polynomial on points of a random low-degree curve passing through the target point. Additionally, the client mixes his queries with noise.

Overall, the client queries come from a distribution that resembles an LSN distribution, namely of noisy samples from a hidden subspace, except here the subspace is not random but rather very structured and consists of permuted low-degree curves. Our security assumption is based on two premises: First, like in LSN, the noise hides the "global" algebraic structure of the queries, and second, the secret permutation obfuscates the "local" low-degree structure of the curves, even when the permutation is reused for multiple retrievals. (See Section 3.1.3 for a more detailed overview.)

We provide preliminary concrete analysis against natural classes of attacks to support our modified design and the proposed choice of parameters. Based on this analysis, we evaluate the security of our schemes against practical attacks. In particular, we conjecture that concretely efficient instances of our framework provide meaningful notions of security even against adversaries that observe a number of queries exceeding the database size by a few orders of magnitude, in some cases up to its square or even more (see Figure 3.1). For security beyond this number of queries, the client can easily "refresh" the database encoding to reset any advantage an adversary might have gained. This can be done either locally after downloading the database, or remotely through a third party. Such a re-encoding procedure takes a relatively short time (see Section 3.6 for more details).

While more investigation is required to gain better confidence in the security of our schemes, our work makes the first steps in what we view as a promising direction.

3.1.2 Detailed Overview

We suggest two kinds of sk-PIR schemes. First, we present a "base scheme", a noiseless special case of the protocol presented in Fig. 2.1, whose conjectured security holds given a bounded number of queries (comparable to the encoded database size). Then, building on the base scheme, we propose a family of augmented schemes that have a slightly higher cost but are conjectured to be asymptotically secure (in a fine-grained sense) given a virtually unbounded number of queries. We identify two types of attacks which we conjecture are most effective against our constructions: *linear-algebraic* attacks, which can be seen as a simpler variant of an algorithm from [CDV21] with degree 1 (Proposition 2.4.2), and *statistical* attacks. The base scheme is vulnerable to an algebraic attack that requires number of samples linear in the database size. The best attacks we know of against the augmented instances, on the other hand, have sample complexity that may be increased by tuning the underlying parameters, to a point of deeming them infeasible in practice.

Note that even the bounded security notion is nontrivial to achieve since the server should be prevented from learning the access pattern (namely, when the same record is read twice). In particular, the server cannot simply store a secretly permuted version of the encrypted database.

The base scheme. The base scheme has optimal download rate and offers a trade-off between server-side storage, online cost and the security of the corresponding augmented schemes. In Table 3.1 we compare four instances of our base scheme, that provide different conjectured levels of practical security as depicted in Figure 3.1, with alternative PIR schemes from the literature.

Each of the instances is characterized by a pair of parameters (m, κ), where m is the number of variables in the polynomial encoding the database and κ is the degree of the curve used for retrieval. (For instance, the first row in Table 3.1 corresponds to $m = 2$ and

$\kappa = 4$.) The choice of (m,κ) affects security (of the augmented schemes) and performance, hence the aforementioned trade-off between the two. Generally speaking, increasing m significantly reduces online cost (see Table 3.1) and increasing κ improves security (see Figure 3.1), both at the expense of more server storage.

Our benchmarks in Table 3.1 refer to an implementation of the base scheme on a standard local machine (a MacBook Pro running macOS 14.5, equipped with 11 CPU cores and 18 GB of RAM). Online runtime includes both client and server computation, but excludes network overhead.

Table 3.1: Our base scheme with various choices of (m,κ) vs. other schemes on a 1GB database. Security evaluation of the different instances is in Figure 3.1.

Scheme	Record Size	Client Storage	Server Storage	Online Runtime (s)	Query Size (MB)	Download Rate
FastPIR [AYA+21]	1 KB	1 MB	≈1×	4.71	1.3	0.0156
XPIR [MBFK16]	288 B	-	≈1×	14.42‡	35.072	0.001
SealPIR [ACLS18]	288 B	3 MB	≈1×	24.14‡	0.31	0.001
OnionPIR [MCR21]	30 KB	5 MB	1×	12.15‡	0.42	0.234
Spiral [MW22]	30 KB	14 MB	1×	3.05	0.22	0.3573
SimplePIR [HHCG+22]	4 KB	123 MB*†	1×	0.131	0.12	0.033
HintlessPIR [LMRSW23]	1B	-	1×	0.666	0.453	0.0003
YPIR [MW24]	1 b	-	≈1×	0.139	0.846	8.13e-5
PIANO [ZPSZ23]	8 B	61 MB†	1×	0.003	0.061	1
MIR [MSR23]	8B	28.3MB†‡	1×	0.002	0.024	0.5
ThorPIR [FLLP24]	1B	1MB†‡	1×	0.0036	0.389	0.0009
Base scheme (2,4)	≥ 17 b	128 b	32×	0.052	0.539	1
(2,5)	≥ 18 b	128 b	50×	0.066	0.68	1
(3,2)	≥ 12 b	128 b	48×	0.0014	0.014	1
(3,3)	≥ 12 b	128 b	162×	0.0021	0.022	1

The client is stateful, i.e. the client storage must be updated in-between queries.

* Client storage is public, i.e. shared for all clients, and can be possibly stored at the server and downloaded before a query.

† Client storage is database-dependant.

‡ Numbers are extrapolated.

We stress that our measures of efficiency, in all the aforementioned aspects, are *non-amortized*. Namely, they reflect the per-query cost regardless of the number of queries. Additionally, our base scheme has download rate *exactly* 1, for any choice of record size larger than $\approx \log n$, where n is the number of records. This should be contrasted with

Table 3.2: Our base scheme vs other schemes in comparable large record-size settings.

Scheme	# Records in DB	Record Size (KB)	Client Storage	Query Size (KB)	Online Runtime (s)	Download Rate
HintlessPIR[LMRSW23]	2^{18}	32	-	1.502	1.39	0.01
OnionPIR[MCR21]	2^{14}	100	5 MB	63	14.38	0.197
SpiralStream[MW22]	2^{14}	100	344 KB	8192	2.38	0.4811
PIANO[ZPSZ23]	2^{20}	100	42 GB*	2.5	0.507*	1
Base Scheme (2,4)	2^{20}	100	128 b	12	1.19†	1

The client is stateful, i.e. the client storage must be updated in-between queries.

* Numbers are extrapolated from the small-record setting.

† Server-side computation is benchmarked in a single-thread setting that computes over the slices sequentially. Runtime can be accelerated by parallel multi-thread computation.

most prior works that consider amortized notions of efficiency.

Our scheme efficiently adapts to large record sizes using standard slicing techniques, with no increase in communication costs and, in particular, while preserving the download rate. We compare our scheme with other practical schemes in the literature designed to support large entries, focusing on the performance metrics of query time and communication cost. The results are summarized in Table 3.2. For further details we refer the reader to Section 3.6.

From bounded to unbounded security. As noted above, we conjecture the base scheme to be secure for up to $(c-1)n$ queries, regardless of the value of m or κ, for a constant $c > 1$ that depends on the choice of parameters (where c is approximately the blow-up in server storage, e.g. $c \approx 32$ for the first entry in Table 3.1). With a bigger number of queries, the server can estimate the number of *distinct* queried records by computing the rank of a sparse matrix defined by the client's messages.

While the above rank attack seems only borderline feasible for our typical parameters,[2] we suggest several simple modifications of the base scheme that eliminate this attack at a low additional cost by injecting a sufficient amount of entropy into the queries. This

[2]E.g., for the first entry in Table 3.1, the matrix has 2^{34} rows with 2^{17} nonzero entries per row (in general, cn rows with 2 to the power of record size non-zeros).

is akin to the LSN conjecture Conjecture 2.1.1, where injecting sufficient noise into low-rank samples disrupts the algebraic structure and effectively neutralizes such linearization attacks. We propose several instantiations of this blueprint:

(i) *The occasional dummy:* A natural strategy is to let the client occasionally make dummy queries. For instance, the client may replace any real query by a dummy one with small probability. Such a solution requires maintaining a small queue of pending queries and is therefore suitable only when the client accesses the database frequently or when delays are tolerable. Alternatively, a client may inject few dummy queries at random time epochs throughout the interaction. The drawback of such a solution is that it is timing-sensitive; dummy queries must not be recognizable. In both solutions, a minimal loss in efficiency is incurred, and the rate remains close to 1.

(ii) *Mixed couples:* A different strategy that is more aligned with the "standard" PIR syntax is to perform, at any PIR query, two invocations of the base protocol; One corresponding to the target database item and another that is dummy, in a uniformly random order. Unlike the first approach, this method doubles the cost per query, mainly in communication. We note that this variant might still be vulnerable to an algebraic attack in the style of [AG11b], albeit one that requires more than n^2 samples. We consider such an attack impractical in our fine-grained setting due to its quadratic sample complexity, which is anyway inferior to existing statistical attacks (see Figure 3.1).

(iii) *Noisy batches:* A third proposal is to apply a variant of the above idea in a setting where multiple PIR queries are batched together. These can be different PIR queries corresponding to different client requests (i.e., indices or retrieved items), or queries corresponding to the same client request, invoked with respect to different slices of a single record in the database. In any such batch of queries, we let the client inject one or more dummy queries at random locations. By carefully balancing the parameters,

Table 3.3: Overhead of different strategies against algebraic attacks, with noise rate μ. For security up to n^c samples, choose $\mu = c/T$, where T is the storage overhead in the underlying base scheme.

Instantiation	Client-to-Server Comm.	Download Rate
Occasional dummy (i)	$\times 1/(1\text{-}\eta)$	$1\text{-}\eta$
Mixed couples (ii)	$\times 2$	0.5
Noisy batches over slices (iii)	$\times 1/\eta^{1\text{-}1/m}$	$1\text{-}\eta$
Noisy batches of queries (iii)	$\times 1/(1\text{-}\eta)^*$	$1\text{-}\eta^*$

* Overhead is amortized over batches of $1/\eta$ queries.

we can obtain download rate that approaches 1 when retrieving a long record or many small records, without relying on timing assumptions.

Accounting for our choice of parameters, given a base protocol with a Tx storage overhead, similarly to Conjecture 2.1.1, we suggest that noise of any rate $\eta \geq 1/T$ for strategies (i) and (iii) is sufficient to eliminate algebraic attacks with linear sample complexity. Furthermore, by Bernoulli's inequality, we know that choosing noise of rate $\eta = d/T$ for a small constant d, e.g. 2 or 3, protects against algebraic attacks with sample complexity up to N^d. We elaborate further in Section 3.4.2 and summarize the costs incurred by the different strategies in Table 3.3.

Worst-case security evaluation. While introducing noise via dummy queries deems algebraic attacks infeasible, potential statistical attacks remain virtually unaffected. We investigate the space of possible statistical attacks against our augmented schemes, in particular attacks that collect *local statistics*. Specifically, we focus on pairwise statistics, i.e. statistics over pairs of client queries, which we conjecture this type of statistical attacks is most effective in our range of parameters.[3] Note that pairwise statistics, in our

[3]Single-query statistics provably do not leak any information, and we conjecture that any statistic of higher locality has lower occurrence.

case, reduces to intersection sizes in random pairs of low-degree curves (passing through a fixed point). We give a concrete evaluation of the query complexity of such attacks against practical instantiations of our framework with respect to various security notions. That is, we generally ask: *How many client queries must an adversary observe to be able to break the security of our schemes?*

First, we consider a worst-case security notion, which is the standard for PIR. Our evaluation for worst-case security is depicted in Figure 3.1, where we quantify security as follows: A protocol is said to guarantee worst-case Q-sample security if an efficient adversary (where we consider any liberal notion of efficiency) cannot distinguish between two sequences of Q PIR queries that correspond to *any* two different sequences of client requests with advantage better than 0.01. [4] Notably, we are not aware of any attack against our schemes, even computationally unbounded, that does not require observing as many queries as Figure 3.1 shows, or breaking an underlying PRP.

We base our evaluation on empirical data and support it by rigorous analysis. Note that under certain parametric choices that give concretely efficient schemes, we conjecture that the best attacks require a number of client queries that exceeds the database size, sometimes by few orders of magnitude, making them irrelevant in practice.

Nevertheless, many of the instances are vulnerable to worst-case attacks with sublinear sample complexity. We argue that even these schemes provide meaningful security in most realistic scenarios.

A more realistic security evaluation. We observe that pairwise-statistics attacks against our schemes are very limited. Specifically, the advantage of these attacks emerges merely from distinguishing between pairs of queries corresponding to a *repeated* client request from pairs of queries corresponding to two *distinct* requests. Consequently, their

[4] The choice of 0.01 advantage is arbitrary. We note that the number of samples required for statistical attacks scales linearly with the resulting advantage. On the other hand, algebraic attacks exhibit a "sharp phase transition", that is, the presented is the number of samples required to obtain any reasonable advantage.

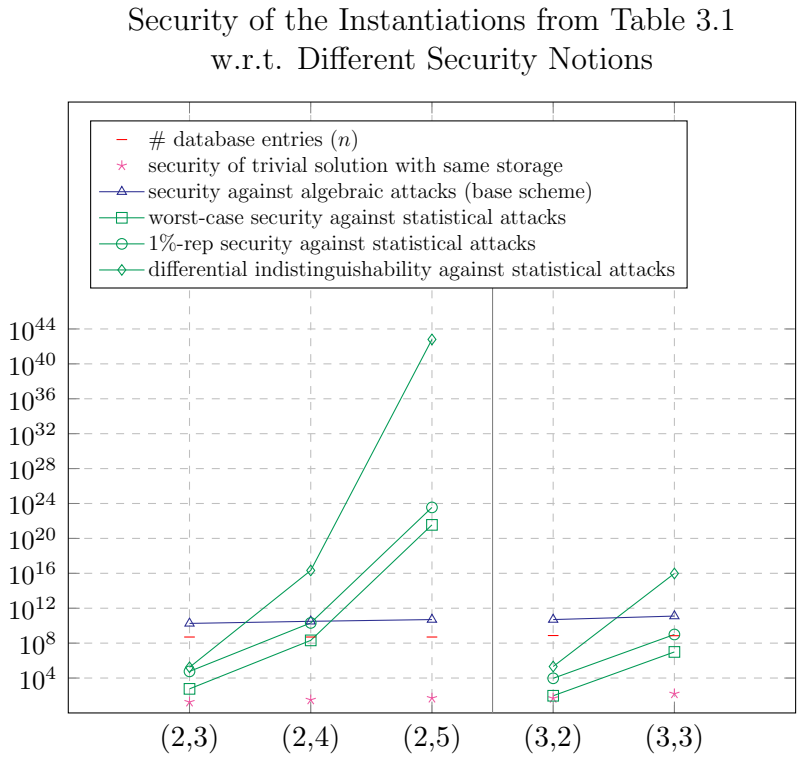

Figure 3.1: The conjectured security of various instantiations of our framework w.r.t. different choices of the parameters (m,κ) and different security notions. Security is quantified by the number of client queries required for distinguishing advantage 0.01 under the respective notion. We look into the following notions: □ worst-case security, that measures indistinguishability between the queries corresponding to any two sequences of client requests, ○ 1%-repetition security, that considers sequences where any request is repeated in at most 1% of the locations, and ◊ differential indistinguishability, where indistinguishability is between any two sequences differing by a single request. We consider two types of attacks: algebraic, which work only against our base scheme, and statistical, that work against the augmented schemes as well. Algebraic attacks require similar number of queries to break any of the above notions of security. Security against statistical attacks is based on extrapolated data (see Figures 3.3 and 3.4 in Section 3.4.2). We compare to the security achieved by the trivial solution with similar storage blow-up (⋆), where the database is replicated, each copy is permuted separately, and copies are accessed in a round-robin fashion. The corresponding attack against the trivial solution breaks all aforementioned notions with advantage 1.

advantage in distinguishing between queries corresponding to two different sequences of client requests is a function of the difference in the number of pairwise repetitions (i.e. pairs of requests to the same database entry) in the two sequences.

In the worst case, the number of pairwise repetitions in a sequence of length L is

approximately $L^2/2$. Indeed, this calculation underlies our worst-case evaluation from Figure 3.1. Note that such a worst case is obtained by a sequence of client requests that is far from realistic, namely a sequence where all requests are to the same database entry.

To gain a better understanding of the trade-off between performance and practical security of our schemes in realistic scenarios, we additionally consider the following relaxed notions of security:

- *α-repetition security,* where we measure indistinguishability between the PIR queries corresponding to any two sequences of client requests with bounded repetition, namely where no database entry is requested in more than an α fraction of the requests. This captures request patterns in many natural scenarios, e.g. when the requests come from a sufficiently entropic distribution or in applications where the client keeps track of highly frequent requests and caches them (locally or at the server).

- *Differential indistinguishability,* which measures indistinguishability between any two sequences that are equal everywhere except at a single request. Roughly speaking, this captures the information leaked about individual requests made by the client. Clearly, differential indistinguishability implies worst-case security, with proportional loss in security, via a hybrid argument. We note that, despite the resemblance, this notion is different than differential privacy, which is sometimes considered for PIR [TDG16, AIVG22, ABG+24].

Evidently, the difference in the number of pairwise repetitions in any two sequences of length L that are considered by the above relaxed notions is much smaller than in the worst case. Between any two α-repetition sequences, the difference in pairwise repetitions is at most αL^2. Between any two sequences differing at a single location the maximal difference is as small as L. Consequently, following our conjecture that pairwise statistics are our "worst enemy", to break the security of our schemes under the relaxed notions requires

observing significantly more queries compared to worst-case security. We demonstrate this in Figure 3.1, where we consider $\alpha = 1\%$ for bounded-repetition security.

Notably, the algebraic attack against the relaxed security of the base scheme still achieves the same advantage given the same number of samples.

3.1.3 Our Techniques

Our starting point is the theoretical sk-PIR candidate from [BIPW17, CHR17]. The high level idea is to encode the database using a secretly permuted version of a standard locally decodable code obtained from Reed-Muller (RM) codes. Concretely, the client's secret key defines both a pseudorandom permutation and a symmetric encryption key. The encoder first applies an RM encoding to the database x, then permutes the resulting codeword, and finally encrypts each entry of the permuted codeword. Recall that in an RM encoding, a database $x \in \mathbb{F}^n$ is extended to an m-variate polynomial p_x over $\mathbb{F}$ which evaluates to x_i at a corresponding point $\mathbf{z}_i \in \mathbb{F}^m$. To privately retrieve x_i, the client picks a random degree-κ curve passing through $\mathbf{z}_i$, and queries p_x on sufficiently many random points on the curve (excluding $\mathbf{z}_i$) to enable recovery of $p_x(\mathbf{z}_i) = x_i$ via interpolation. The intuition behind security is that the secret permutation obfuscates the structure of low-degree curves corresponding to the client requests, making them indistinguishable from permuted uniformly random curves.

Note this is a special case of the construction discussed in Chapter 1, where client queries correspond to codewords from the dual of a random code. Here, the geometry underlying Reed-Muller codes allows us to sample dual codewords as evaluations over algebraic low-degree curves. Each such codeword has an extremely low Hamming weight – supported only at q positions – and we reveal only the support rather than the actual values. The sparsity of such codewords is the key to the the low communication cost of our protocols. Security relies on the indistinguishability of these low-weight codewords from arbitrary low-weight vectors under the secret permutation, which masks their structure

that would otherwise reveal the client's queries.

Fixing $m \geq 2$ to be constant, the above construction requires the client to read $r = O(\kappa \cdot n^{1/m})$ entries of the encoded database. However, it has several sources of inefficiency. First, the constraints on the size of $\mathbb{F}$ imply that the encoding size must be at least $\kappa^m \cdot n$, a prohibitively large overhead for typical choices of a cryptographic security parameter κ. Second, the download rate is $1/r$, which implies very high communication for large records. Finally, the concrete computational cost of polynomial interpolation is quite high, which may significantly impact the client's efficiency.

Our base scheme incorporates three different ideas to mitigate the above bottlenecks.

First, via a more refined security analysis, we observe that for sufficiently large databases we can choose the degree parameter κ to be as small as 3 or even 2. This corresponds to a "fine-grained" security setting, where a small power of the database size can be considered an infeasible quantity.

Second, to eliminate the high concrete cost of polynomial interpolation, we use a careful choice of the interpolation points that makes all of the Lagrange coefficients equal to 1. In other words, the target value x_i is reconstructed from the queried values of p_x by simply adding them up.

Finally, the above change allows us to compress the server's response by having the server send only the sum of the queried entries. Here we observe that if encryption of the permuted codeword entries is done via additive masking with the output of a pseudorandom function, then the client can do the unmasking.

Since we make the Lagrange interpolation coefficients public, our base scheme is exposed to a simple "linearization" attack that was already observed in [BIPW17, CHR17]. However, as discussed above, this attack is often impractical with our choice of parameters due to its high query complexity and computational cost. Moreover, this vulnerability can be circumvented by adding dummy queries in what we referred to above as our "augmented schemes". This defense mechanism is similar in spirit to an alterantive sk-PIR

candidate construction from Canetti et al. [CHR17], which reveals the Lagrange coefficients but mixes the points of the curve with additional dummy points.

Security assumptions. The security of our base construction boils down to the hardness of distinguishing a "planted" distribution over low-degree curves over a finite field from the uniformly random distribution over such curves, when the domain undergoes re-labeling via a random permutation. Specifically, in the planted distribution, the curves are sampled at random conditioned that they pass through points planted by the adversary. In our more resilient proposals, the adversary sees a *noisy* planted distribution, namely when the distribution outputs a uniformly random curve, instead of planted, with some small probability.

We define the *permuted low-degree with noise* problem (or PLDN for short), which captures the hardness in breaking our schemes, and formalize it in Definition 3.1.1 below. The PLDN problem is inspired by the *permuted puzzle* framework, which was previously introduced and studied in the context of sk-PIR [BIPW17, BHMW21], and most closely resembles an alternative "noisy" variant of the permuted low-degree curves assumption from [CHR17].

Definition 3.1.1 (PLDN conjecture)**.** *Let $m, \kappa \in \mathbb{N}$ be fixed parameters. For any prime power $q \in \mathbb{N}$, let $\mathbb{F} := \mathbb{F}_q$ be the finite field of size q.*

*The $(m, \kappa, \eta, T, \varepsilon)$-*permuted low-degree with noise *conjecture (PLDN for short) asserts that no oracle-aided algorithm that makes $T(q)$ queries to its oracle is able to distinguish between samples from the oracle PLDN^{π}, defined below, with adversarially chosen inputs α (i.e. the* planted distribution*), and samples from the oracle PLDN^{π} with uniformly random i.i.d. inputs $\alpha \leftarrow \mathbb{F}^m$ (i.e. the* random distribution*), with advantage better than $\varepsilon(q)$, when π is a random permutation over $\mathbb{F}^m$.*

- $\mathsf{PLDN}^{\pi}(\alpha \in \mathbb{F}^m)$*:*

1. *With probability $\eta(q)$, output $q-1$ uniformly random points in $\mathbb{F}^m$ (with replacement).*
2. *Otherwise, sample a uniformly random curve C of degree at most κ over $\mathbb{F}^m$, such that $C(0) = \alpha$.*
3. *Output $\pi(C(t))$ for all $t \in \mathbb{F} \setminus \{0\}$, in a uniformly random order.*

Remark (LSN interpretation). One can equivalently state PLDN in the language of the LSN assumption from Chapter 2 Definition 2.1.1. Here, the adversary sees samples from a *noisy dual code* of exact sparsity q—that is, vectors of Hamming weight q with all nonzero entries equal to 1—versus uniformly random vectors of the same weight. Even with up to $T(q)$ oracle queries, no efficient distinguisher can tell these two cases apart with advantage better than $\varepsilon(q)$.

This remark shows that PLDN is just the "curve" instantiation of learning a noisy sparse code from Chapter 2, specialized to the Reed–Muller dual and weight-q queries. Although PLDN introduces new notation specific to Reed–Muller curves, it mirrors the LSN assumption's core principle: in both cases the adversary must distinguish *structured but obfuscated* queries from truly random ones under noise and permutation.

Asymptotically, for any $m, \kappa > 1$ and $\eta > 1/m!\kappa^m$, we conjecture that $(m, \kappa, \eta, T_{\mathsf{stat}}, \varepsilon)$-PLDN holds for $T_{\mathsf{stat}} = o(q^{\alpha \cdot m\kappa - 1})$ and $\varepsilon = o(q^{-\alpha \cdot m\kappa - 1})$, for any $\alpha < 1/2$. In the noiseless regime, where $\eta = 0$ and algebraic attacks are relevant, we conjecture that $(m, \kappa, 0, T_{\mathsf{alg}}, \varepsilon)$-PLDN holds for $T_{\mathsf{alg}} = \min(o(m!\kappa^m q^m), T_{\mathsf{stat}})$ and the same ε. Our asymptotic conjectures are based on rigorous analysis of the attacks we conjecture are most successful (i.e. linear algebraic attacks and pairwise-statistics attacks).

Our concrete security evaluation in Fig. 3.1 is derived from empirical experiments analyzing the success of the aforementioned attacks in the corresponding fixed regimes of parameters, including some small parameter settings where our conservative asymptotic conjectures, emerging from loose analysis in these regimes, are obsolete.

For more details regarding our security conjectures, both in the asymptotic and concrete regimes, and the underlying rigorous and empirical evidence, we refer the reader to Section 3.4.2 and Appendix .2.

Lastly, we note that we conjecture hardness of PLDN holds even w.r.t. certain noise distributions that contain some structure, and are not Bernoulli i.i.d., as long as they produce a similar noise rate η. This allows us to argue security of some of our proposals for augmented schemes, namely the *mixed couples* strategy (ii) and *noisy batches* (iii).

3.2 Preliminaries

Notation. We use n to denote the number of elements in a database and N to denote the number of elements in a database encoding. We consider databases where entries are "sliced" into s parts, with each part consisting of a value from a pre-specified finite field $\mathbb{F}$. We take $s = 1$ by default.

We denote $[\ell] = \{1, \ldots, \ell\}$ for any $\ell \in \mathbb{N}$. For a polynomial P over a finite field, we let $\deg P$ denote the total degree of P. We typically use boldface, e.g. $\boldsymbol{\alpha}$, to denote a vector (typically of field elements) and use $\boldsymbol{\alpha}_i$ to denote the i^{th} element in such a vector. Lastly, for a finite field $\mathbb{F}$, we denote $\mathbb{F}^* = \mathbb{F} \setminus \{0\}$.

3.2.1 Reed-Muller Codes

Reed-Muller codes are a well-studied class of error-correcting codes that have been widely used in digital communication systems. They are known for their efficient decoding algorithms and favorable error-correction properties. Recently, there has been interest in utilizing Reed-Muller codes for cryptographic applications, specifically in the construction of Oblivious LDCs for PIR protocols [BIPW17, CHR17].

Generally, a Reed-Muller code is defined over a finite field $\mathbb{F}$ and consists of all evaluations of multi-variate low-degree polynomials over $\mathbb{F}$.

Definition 3.2.1 (Reed-Muller Codes)**.** *Let q be a power of prime and let $\mathbb{F}$ be the finite field of size q. Let $d, m \in \mathbb{N}$. The* degree-d m-variate Reed-Muller code *(RM-code for short) is an $[N = q^m, n = \binom{m+d}{m}]$-code over $\mathbb{F}$ defined as*

$$\mathsf{RM}_q(d,m) = \{(P(\alpha))_{\alpha\in\mathbb{F}^m} \mid P \in \mathbb{F}[\alpha_1 \dots \alpha_m], \deg(P) \le d\}.$$

Useful to us is the fact that Reed-Muller codes exhibit efficient and systematic encoding functions. The constructions presented in the paper require such an encoding function but are otherwise immaterial to its choice. Therefore, we implicitly assume an arbitrary choice of an efficient and systematic encoding function underlying any RM-code. In a context where an RM-code $\mathsf{RM}_q(d,m)$ is implicit, we denote, for any $x \in \mathbb{F}^n$, its encoding under the corresponding function by $P_x \in \mathsf{RM}_q(d,m)$. Further, we denote, for any $i \in [n]$, by $\widehat{i} \in \mathbb{F}^m$ the field element satisfying $P_x(\widehat{i}) = x[i]$ for all x (such an $\widehat{i}$ exists since the encoding function is systematic).

In particular, systematic encoding allows for a local decoding procedure that recovers the value of the data at a given location by reading the encoded data along a curve. First, let us recall the definition of a curve.

Definition 3.2.2 (Algebraic Curve)**.** *Let $\kappa, m \in \mathbb{N}$ and let $\mathbb{F}$ be a field. A* degree-κ curve C over $\mathbb{F}^m$ *is defined by m univariate polynomials $p_1, \dots, p_m$ as the following list of points in $\mathbb{F}^m$,*

$$C = \langle\ (p_1(t), \dots, p_m(t))\ \rangle_{t\in\mathbb{F}}.$$

We slightly overload notation and use C to also denote the mapping induced by the curve, namely

$$C : t \mapsto (p_1(t), \dots, p_m(t)).$$

The following is a consequence of standard polynomial interpolation and is used in

the PIR schemes from [BIPW17, CHR17].

Proposition 3.2.1 (Decoding an RM-Code via a Curve)**.** *Let q be a power of prime and $\mathbb{F}$ be the finite field of size q, and let $d, m \in \mathbb{N}$ and $N = q^m$ and $n = \binom{m+d}{m}$. Let $x \in \mathbb{F}^n$ and let $P_x \in \mathbb{F}^N$ be its (systematic) encoding under $\mathsf{RM}_q(d, m)$. Let $\kappa \in \mathbb{N}$. For any degree-κ curve C over $\mathbb{F}^m$ such that $\widehat{i} \in C$, the value of $x[i]$ is uniquely (and efficiently) determined by any $(d\kappa + 1)$-size subset of $\{P_x(c) \mid c \in C\}$.*

3.3 Background: PIR via Reed-Muller

We recall the outline of the sk-PIR constructions from [BIPW17, CHR17] which are based on Reed-Muller codes (recall Definition 3.2.1).

Their constructions are in the setting where the client holds a secret key and the server stores an encoding of the database that is prepared by the client, possibly in a pre-processing phase. More specifically, the database is viewed as a vector $x \in \mathbb{F}^n$ and is encoded via a RM-code $\mathsf{RM}_q(d, m)$ for some $d, m \in \mathbb{N}$ such that $\binom{m+d}{m} \geq n$ (where q is the size of $\mathbb{F}$). Then, the resulted encoding, which we denote by $P_x \in \mathbb{F}^N$, for $N = q^m$, is first encrypted (entry-by-entry), then permuted by a pseudorandom permutation PRP over $[N]$. The result is stored in the server. The client's secret key k consists of the key to PRP and the symmetric key used to encrypt the database entries.

To retrieve the database value at location $i \in [n]$, the client performs the following steps:

1. Sample a degree-κ curve C that passes through $\widehat{i}$, where $\widehat{i} \in \mathbb{F}^m$ satisfies $P_x(\widehat{i}) = x[i]$ (see Section 3.2.1).

2. Query the server at a random $(d\kappa + 1)$-size subset of $\{\mathsf{PRP}_k(c) \mid c \in C \setminus \{\widehat{i}\}\}$.

3. Interpolate to obtain $P_x(\widehat{i})$.

Recall interpolation at the last step is possible due to Proposition 3.2.1. Notice that the above outline implicitly requires that $d\kappa + 1 < q$. Otherwise, κ can be arbitrarily chosen yet its choice plays a significant role in the security of the protocol, as we shall see.

3.3.1 Complexity

Before reasoning about the security of the scheme, however, let us first analyze its complexity. The encoding of a database of size n over $\mathbb{F}$ consists of $N = q^m$ field elements. Since we require $\binom{m+d}{m} \geq n$ and, consequently, $d^m \geq n$, the encoding incurs a blow-up of $q^m/n > (d\kappa)^m/n \geq \kappa^m$ in server-side storage. To retrieve a database element $x[i] \in \mathbb{F}$, the client sends the server a total of $d\kappa + 1$ locations in $[N]$ and receives $d\kappa + 1$ corresponding field elements. When our database contains larger entries, say vectors in $\mathbb{F}^\ell$ (or, equivalently, when each database entry is "sliced" into ℓ field elements), one can apply ℓ parallel instantiations of the PIR scheme on each of the ℓ "slices" of the database, using the same randomness, to amortize the length of the client's message. Note that this, however, does not affect the download rate since the server still has to respond with the same number of field elements per slice.

To conclude, the basic PIR scheme from [BIPW17, CHR17], when instantiated with parameters $\kappa, m \in \mathbb{N}$ over a database of size n, where each entry contains an element in $\mathbb{F}^\ell$, has

- at least κ^m blow-up in server-side storage,
- client message of length $\approx m\kappa \cdot n^{1/m} \log |\mathbb{F}|$ bits, and
- download rate $\approx \kappa \cdot n^{1/m}$.

Lastly, note that server-side computation is trivial, and the server acts simply as a RAM machine.

3.3.2 Security

Intuitively speaking, the security of the above outline relies on the premise that the pseudorandom permutation over $\mathbb{F}^m$ sufficiently obfuscates the algebraic structure of the underlying low-degree curves.

This may be captured, in particular, by the following distinguishing game between two types of samples: In one scenario, each sample is a random subset of a uniformly random low-degree curve that passes through a point of the adversary's choice. In another scenario, the samples are uniformly random subsets of matching size.

Both of Canetti *et al.* [CHR17] and Boyle *et al.* [BIPW17] consider the hardness of the above distinguishing game as the basis for the security of their respective schemes. The former work, however, mainly focuses on a potentially harder variant where additional "noise" is added to the responses of the oracle, consequently calling it the *hidden permutation with noise* problem. In this work, we choose to roughly follow the terminology from [BIPW17] since, in particular, our practical scheme (presented in Section 3.4) is not compatible with the noisy version deifned in [CHR17].

Definition 3.3.1 (Permuted Low-Degree Puzzle [BIPW17])**.** *Let $m \in \mathbb{N}$ be a fixed dimension parameter. For any prime power $q \in \mathbb{N}$, let $\mathbb{F} := \mathbb{F}_q$ be the finite field of size q and let $\kappa : \mathbb{N} \to \mathbb{N}$ and $\ell : \mathbb{N} \to \mathbb{N}$ be such that $\kappa(q) < \ell(q) < q$.*

The permuted low-degree puzzle *with parameters m, κ, ℓ, shorthanded (m, κ, ℓ)-PLD, consists of distinguishing between two oracles: $\mathsf{LowDeg} = \{\mathsf{LowDeg}_q^\pi\}$, parameterized by a permutation $\pi : \mathbb{F}^m \to \mathbb{F}^m$, and $\mathsf{Random} = \{\mathsf{Random}_q\}$. Both oracles take as input a vector in $\mathbb{F}^m$ and output $\ell := \ell(q)$ such vectors, and are defined as follows.*

- $\mathsf{LowDeg}_q^\pi(\alpha)$*:*

  1. *Sample a uniformly random degree-$\kappa(q)$ curve C over $\mathbb{F}_q^m$ such that $C(0) = \alpha$.*
  2. *Sample uniform and distinct $t_1, \ldots, t_\ell \leftarrow \mathbb{F}_q \setminus \{0\}$.*

3. For $i = 1, \ldots, \ell$, let $\mathbf{c}_i = C(t_i)$ and output $\mathbf{w}_i = \pi(\mathbf{c}_i)$.

- $\mathsf{Random}_q(\alpha)$*: output uniformly random and distinct* $\mathbf{w}_1, ..., \mathbf{w}_\ell \leftarrow \mathbb{F}^m$.

We say that an oracle-aided algorithm $\mathcal{A}$ is a (T, S, Q, ε)-solver *of* (m, κ, ℓ)*-PLD if, for any $q \in \mathbb{N}$, $\mathcal{A}$ runs in time $T(q)$, uses space $S(q)$ and makes at most $Q(q)$ queries, and its advantage in distinguishing between* Random *and* LowDeg^π *for a random $\pi \leftarrow S_N$, namely*

$$|\Pr[\mathcal{A}^{\mathsf{LowDeg}^\pi}(1^q) = 1] - \Pr[\mathcal{A}^{\mathsf{Random}}(1^q) = 1]|,$$

is at least $\varepsilon(q)$. Lastly, we say that a PLD puzzle is (T, S, Q, ε)-hard *if it has no non-uniform (T, S, Q, ε)-solver.*

The above puzzle is a special case of a generalized framework of *permuted puzzle problems* that was subsequently developed by Boyle, Holmgren and Weiss [BHW19b].

Boyle *et al.* [BIPW17] conjecture that no adversary is able to solve the permuted low-degree puzzle with advantage that exceeds a negligible function in κ when $\ell < q - 1$.[5]

Conjecture 3.3.1 (Conjecture 4.2 in [BIPW17])**.** *For any $m \in \mathbb{N}$ and $\kappa, \ell : \mathbb{N} \to \mathbb{N}$ such that $\ell < q - 1$, the (m, κ, ℓ)-PLD puzzle is (T, S, Q, ε)-hard for any polynomials T, S, Q and ε that is inverse-polynomial in κ.*

The security of their sk-PIR scheme reduces to Conjecture 3.3.1. In particular, κ constitutes a "security parameter" in their construction and, since they aim for standard polynomial security, it is chosen to be asymptotically large (i.e. super-logarithmic in the least).

[5] Although in Conjecture 4.2 in [BIPW17] ℓ is required to be only smaller than q, their work suggests a linearization attack that in fact breaks the conjecture, when $\ell = q - 1$, as we discuss later in Section 3.4.

3.4 Our Base Scheme

We hereby present our base scheme. Recall our base scheme is vulnerable to linear-algebraic attacks that require observing a number of queries linear in n. For various strategies for upgrading the base scheme to variants that withstand these attacks, we refer the reader to Section 3.1.2.

Our base construction follows the framework from [BIPW17, CHR17] of building sk-PIR based on Reed-Muller codes (described in Section 3.3). To obtain a practical protocol, we introduce a couple of adjustments to the construction by Boyle *et al.* [BIPW17], that dramatically improve concrete efficiency on two dimensions: server-side storage and download rate.

Extremely Low Degree. First, we set the degree of the curves, i.e. κ, to be a small constant (say, between 2 and 5). When κ is chosen as such, the blow-up in server-side storage reduces drastically compared to the original proposals from [BIPW17, CHR17] to become merely a proportionally small constant (see details in Section 3.3.1). Importantly, κ must be chosen to be at least 2 due to a simple attack in the case where $\kappa = 1$ (see Appendix .5).

An immediate consequence of such a choice of κ is a polynomial-time solver of the corresponding permuted low-degree puzzle (see Definition 3.3.1). In particular, a birthday attack solves the puzzle given $\approx q^{m\kappa/2-1}$ samples from the oracle (see Appendix .4 for more details), which is polynomial in the size of the database. In particular, this places us in the fine-grained regime, where the best we can hope for is security against adversaries with capacity bounded by a concrete polynomial.

Download Rate 1 via Oblivious Interpolation. Second, we propose a new method to reduce the download rate of the protocol all the way to 1, in the expense of lightweight

sub-linear server-side computation.

At the heart of our method is a simple linear interpolation function for retrieving the value at the target location, sought by the client, given all other values on the random curve returned by the server. Crucially, the interpolation is *oblivious* in the target location. In fact, it is simply a summation and, in particular, it can be performed by the server. Note, however, since the server sees only encryptions of the values in the RM codeword, we must be careful in choosing the underlying symmetric encryption scheme to allow for linear homomorphism, i.e. where the sum of ciphertexts can be decrypted to the sum of their corresponding plaintexts. Overall, the client sends the locations of *all* points in the curve except for the target location (similarly to the case where $d\kappa + 1 = q - 1$ in the original scheme) and, in return, receives a single element in $\mathbb{F}^m$ corresponding to their sum.

The above optimization introduces a vulnerability in security via a linearization attack that was described in [BIPW17, CHR17]. Recall that Conjecture 3.3.1 requires the sampled subset is of size $\ell < q-1$. This comes for a reason. By setting $\ell = q-1$ and allowing for oblivious interpolation, we are creating a situation where the requested database item may be interpolated using public interpolation coefficients (this is essentially what allows us to move interpolation to the server side). Roughly speaking, notice that the vector v of interpolation coefficients (embedded as a vector in $\mathbb{F}^{q^m}$) always belongs to a co-set of the kernel of the (permuted) RM-code which corresponds to the requested database value; this is because $v^T \cdot c = c_i$ for any RM-codeword c (i.e. database encoding). This implies that client queries to the same database element come from a low-rank subspace and, therefore, can be distinguished from random given sufficiently many samples according to the analysis in Chapter 1.

Despite the above attack, we can still utilize our optimal-rate protocol for sk-PIR. First, the attack works only given $c \cdot n$ queries corresponding to less than n distinct database items. Under any of our parameter choices, c is at least 31. Therefore, one

can still argue a fine-grained notion of security against adversaries who do not observe sufficiently many queries to non-distinct items. Second, the attack completely fails under any of our extended protocols which we describe in Section 3.1.2, where we add dummy queries to create noise in the algebraic structure.

3.4.1 The Protocol

We now present our construction in detail.

Construction 3.4.1. *Our sk-PIR scheme is over a database that contains $n := n(\lambda)$ entries, where each entry is an element of a field $\mathbb{F}$ of size $q := q(\lambda)$. The scheme encodes the database via a Reed-Muller code (recall Definition 3.2.1) and, therefore, it presumes that $n \leq \binom{d+m}{m}$ for parameters $m \in \mathbb{N}$ and $d := d(\lambda)$. Recall the existence of an efficient and systematic encoding of any RM-code and the corresponding mapping between any location $i \in [n]$ in the data to a location $\widehat{i} \in \mathbb{F}^m$ in its encoding where the same value is found (see Section 3.2.1 for more details). Additionally, let $\kappa \in \mathbb{N}$ be such that $q > d\kappa + 1$.*

The construction assumes the existence of a pseudorandom permutation family $\mathsf{PRP} = \{\mathsf{PRP}_\lambda : \mathbb{F} \to \mathbb{F}\}$ and a pseudorandom function family $\mathsf{PRF} = \{\mathsf{PRF}_\lambda : \mathbb{F}^m \to \mathbb{F}\}$.

- $\mathsf{G}(1^\lambda)$*: Sample $\pi \leftarrow \mathsf{PRP}_\lambda$ and $g \leftarrow \mathsf{PRF}_\lambda$ and output $\mathsf{sk} = (\pi, g)$.*
- $\mathsf{E}(\mathsf{sk} = (\pi, g), x)$:
  1. *Let $P_x \in \mathsf{RM}_q(d, m)$ be a systematic Reed-Muller encoding of x and let $\widetilde{P_x}$ be its permutation under π. That is, for every $\alpha \in \mathbb{F}^m$, set*

$$\widetilde{P_x}(\pi(\alpha)) = P_x(\alpha).$$

  2. *Encrypt $\widetilde{P_x}$ using the PRF g to obtain the encoded database X. Specifically, for every $\alpha \in \mathbb{F}^m$, let*

$$X[\alpha] = \widetilde{P_x}(\alpha) + g(\alpha).$$

3. Output X.

- $\mathsf{Q}(\mathsf{sk} = (\pi, g), i)$*:*

  1. Sample a uniformly random degree-κ curve C over $\mathbb{F}^m$ such that $C(0) = \widehat{i}$.

  2. Sample a uniformly random permutation $\pi' : [q-1] \to \mathbb{F}^*$ and output

$$Q = (\pi(\mathbf{c}_{\pi'(1)}), \ldots, \pi(\mathbf{c}_{\pi'(q-1)})),$$

  where $\mathbf{c}_j = C(j)$.

- $\mathsf{A}(X, Q)$*: Output $a = \Sigma_{\mathbf{q}\in Q} X[\mathbf{q}]$.*

- $\mathsf{D}(\mathsf{sk} = (\pi, g), i, a)$*: Output $\Sigma_{\mathbf{q}\in Q} g(\mathbf{q}) - a$.*

Correctness. Correctness of the protocol relies on the following well-known algebraic fact.

Fact 3.4.1. *Let $q \in \mathbb{N}$ be a prime. For any polynomial P over a finite field $\mathbb{F}$ of size q, it holds that $\sum_{x\in\mathbb{F}} P(x) = 0$.*

The above implies that RM-codeword entries on a curve sum to 0 since they form all evaluations of a univariate polynomial. This, in turn, implies the correctness of our oblivious interpolation method.

Complexity. The complexity of our sk-PIR scheme is a function of multiple parameters. These include the field size q and the number of variables that define the underlying RM-code, and the degree κ chosen for the curves used for decoding. We obtain the following complexity measures given any choice of q, m and κ (recall terminology in Definition 2.3.3).

– Storage-rate $m!\kappa^m$. The server stores an RM-encoding of the database. For a database of size n, we must choose the parameters for the RM code to satisfy

$n = \binom{m+d}{m}$ (see Definition 3.2.1). When $m \ll d$ is a small constant, this implies $n \approx d^m/m!$. Additionally, recall that we require $q > d\kappa + 1$ for correct decoding (see Proposition 3.2.1). The resulted RM-encoding is consequently of size $N = q^m \approx (d\kappa)^m \approx (m!\kappa^m) \cdot n$.

– Query-length $(q-1) \cdot m \log q$. The client sends a list of $q-1$ points in $\mathbb{F}^m$, consisting of all points on a low-degree curve, except for the evaluation at zero.

– Download-rate 1. The server sends in return only a single entry (i.e. a field element) to the client.

Efficiency. Compared to the random-linear-code instantiation analyzed in Chapter 2 Fig. 2.1 — where each query is a full-length dual-space vector of weight $\Theta(N)$ — our RM-based queries have weight only q, and every nonzero entry equals 1. Concretely, instead of transmitting an N-bit query, the client need only send the list of q support positions (each encoded in $m \log q$ bits), reducing query communication from $\Theta(N)$ down to $O(q\, m \log q)$. Moreover, the RM structure yields a simple and efficient method to generate these sparse dual codewords: one picks a random degree-κ polynomial, evaluates it at all nonzero $t \in \mathbb{F}$, and applies the secret permutation. This structure not only slashes bandwidth but also yields very fast client-side sampling compared to generic dual-space queries. Furthermore, on the server side the work reduces to fetching those q positions and summing their (encrypted) values—an $O(q)$ addition—rather than performing a full matrix-vector multiplication over $\mathbb{F}^N$ as in the base protocol Fig. 2.2.

3.4.2 Cryptanalysis

Recall the security of our base scheme and its augmented variants (that involve dummy queries) is derived from the hardness of the permuted low-degree with noise problem (PLDN), defined in Definition 3.1.1. We perform preliminary cryptanalysis to establish

confidence in the hardness of the PLDN problem, particularly when the parameter κ is a small constant (as opposed to the original proposals from [BIPW17, CHR17]). We consider a number of natural attack strategies which we divide into two main categories: algebraic attacks and statistical attacks. We hereby provide an overview, and refer the reader to Appendix .2 for an elaborate account.

Algebraic Attacks. The base scheme is susceptible to a linearization attack, which exploits the fact that $\mathbb{F}^N$-vectors representing the client's queries to the same database item come for a low-rank subspace. Specifically, such vectors are in the same coset of the kernel associated with the RM-code underlying the construction. Such a subspace exhibits rank of at most $N - n$, where $n \approx N/(m!\kappa^m)$ is the maximal dimension such an RM-code can have. In fact, even when the sequence of PIR queries corresponds not only to a single database item, but to few, the algebraic rank of the query vectors is smaller than full, i.e. N. Specifically, the rank produced by a sequence of vectors containing queries corresponding to n' distinct database elements is at most $N-n+n'$ (each distinct database element adds a different coset to the subspace).

The above suggests the following attack against our scheme: Collect samples of PIR queries until a linear dependency is found. The number of samples required to create a linear dependency gives good indication regarding the rank of the subspace these samples come from (see Proposition 3.4.1). By the above observation, this rank gives away the number of distinct database elements that were requested throughout the sequence of queries.

Proposition 3.4.1. *The probability that t samples from a rank-r linear subspace are linearly independent is $\approx e^{-q^{-(r-t-1)}}$.*

By Proposition 3.4.1, to distinguish between random samples from a rank-$(N-n)$ space and uniformly random samples, an attacker must observe $t \approx N-n$ samples to obtain any reasonable advantage. This precisely parallels the degree-1 case of the attack

algorithm described in [CDV21], where for a dual subspace of dimension k, an algebraic (rank) attack likewise requires about $n-k$ samples to succeed. In our scheme, however, the samples are not uniform over the low-rank subspace, but rather come from a structured distribution. Specifically, they represent random degree-κ curves over $\mathbb{F}^m$. Despite this, we conjecture that such a distribution demonstrates a similar behavior in this regard and, consequently, conjecture that our base scheme is secure against adversaries that observe less than $N - n$ client queries. We support this conjecture with empirical evidence, as shown in Fig. 3.2.

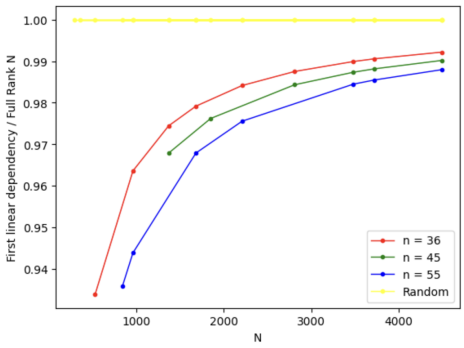

Figure 3.2: Ratio between appearance of first linear dependency and N, where n is DB size and N is the size of its RM encoding. For queries corresponding to uniformly random database entries, dependencies appear at the $(N+1)^{th}$ sample, whereas when the samples correspond to a repeated client request, the first dependency appears at the $(N-n+1)^{th}$ sample.

.

Algebraic Attacks against Augmented Schemes. The rank attack described above relies on the algebraic structure satisfied by client queries in the base scheme. In the augmented schemes, we disturb this algebraic structure by mixing the queries with noise at a certain rate $\eta > 0$. With LSN conjecture, this modification significantly increases the sample complexity of algebraic attacks, already with relatively small noise rate given the dual space dimension ratio c.

Roughly speaking, an algebraic attack against our noisy schemes attempts, given $\gg n$ samples, to tell whether a (hidden) majority of them, corresponding to the noiseless

samples, comes from a rank-n subspace. The simple rank attack against the base scheme still works as long as the noise rate is not sufficiently high for the noisy samples to have full rank. Thus, to eliminate the straight-forward rank attack with linear sample complexity, we require $\eta > 1 - n/N$, where N is the full rank of the domain. This is indeed the threshold above which the expected rank of the noisy samples becomes full with overwhelming probability.

What happens when $\eta > 1 - n/N$? While linear dependencies among the noisy samples are reduced to trivial in this regime, non-trivial low-degree dependencies might still appear. For instance, consider the regime where $1-n/N < \eta \ll 1-(n/N)^2$. Take N^2 samples that correspond to the same client query. Among these, there exist more than n^2 noiseless samples at average. These noiseless samples come from a rank-n subspace and, through their linear dependency, exhibit in particular a degree-2 dependency (namely they are all roots of some degree-2 polynomial). Such a degree-2 dependency is observed in random N^2 samples with very low probability (this is also the case when the N^2 samples correspond to different client queries). Such an attack, which is similar in spirit to algorithms from [AG11b, CDV21], may be generalized to any low degree c, requiring $\eta < 1 - (n/N)^c$ and N^c samples. We conjecture that this approach captures the best attainable algebraic attacks and, by this heuristic, derive our choice of noise rate described in Table 3.3.

Statistical Attacks. When it comes to statistical attacks, since the database encoding is permuted under a pseudorandom permutation, it only makes sense to consider statistics that are symmetric in the field elements. We look at simple classes of such attacks: a birthday attack and pairwise-statistics attacks. We hereby survey our main cryptanalytic efforts in this respect, and discuss further in Appendix .2.

Preliminary analysis shows that the number of all possible PIR queries corresponding to a fixed client request, equivalently all possible degree-κ curves that pass through the

associated fixed point, is at most $q^{m\kappa-2}$ (see Propositions .3.1 and .3.2). Empirical data indicates that this bound is tight (see Figure 8). Consequently, we conjecture that a birthday attack, which relies on finding a collision in the space of all possible queries, requires observing at least $O(q^{m\kappa/2-1})$ samples (see Proposition .4.1).

Next, we look into attacks that collect local statistics over the sequence of queries, i.e. statistics involving single queries, pairs of queries and possibly larger subsets. Via a simple rigorous argument (a proof is in the full version),it can be shown that looking at individual queries cannot possibly convey any information about the underlying client requests, assuming the underlying PRP π is random. The next thing to analyze, then, is information leaked by pairwise statistics, which essentially reduces to the intersection size between two queries. In fact, we conjecture that such statistics are most effective among local-statistics attacks, as we believe that statistics with higher locality occur less frequently (in total along a sequence of queries).

To empirically bound the advantage gained by pairwise statistics, we estimate the squared Hellinger distance between the intersection size distribution exhibited by a pair of queries corresponding to the same client request vs. a pair of queries corresponding to different client requests. To that end, we perform the following experiment: We sample t degree-κ curves over $\mathbb{F}^m$ that pass through a fixed point (say 0^m), record the occurrence of any intersection size among all $\binom{t}{2}$ pairs, through which we calculate, for any s, the probability that a random pair of queries has intersection size s. We repeat this with uniformly random degree-κ curves to obtain an estimate for the probability that a pair of random curves has any given intersection size. Given an estimation of the pdf of the two intersection-size distributions, we calculate the squared Hellinger distance δ^2. This gives a bound on the number of samples required for an attack: to gain advantage α, the adversary must observe $\alpha^2/\sqrt{2}\delta^2$ pairs, which requires $2^{1/4} \cdot \alpha/\delta$ samples.

Due to the obvious limitation in computation power, we carry the aforementioned experiments with respect to small values of q, that are much smaller than the values

used in our proposed schemes from Table 3.1 (see Table 3.4). To obtain an evaluation for the settings in our schemes, we extrapolate under the hypothesis that the distance is a function of the form $\alpha \cdot q^{\beta}$ for $\alpha, \beta > 0$. This is solidified by the empirical outcome for small q values. The results of the "small" experiments are presented in Figure 3.3. Figure 3.4 depicts the extrapolated values relevant to our schemes.

Following the calculation described above, we obtain our heuristics in Figure 3.1. Notice that our experiments show the distance decays exponentially with κ, m and $\log q$, in alignment with our asymptotic conjectures.

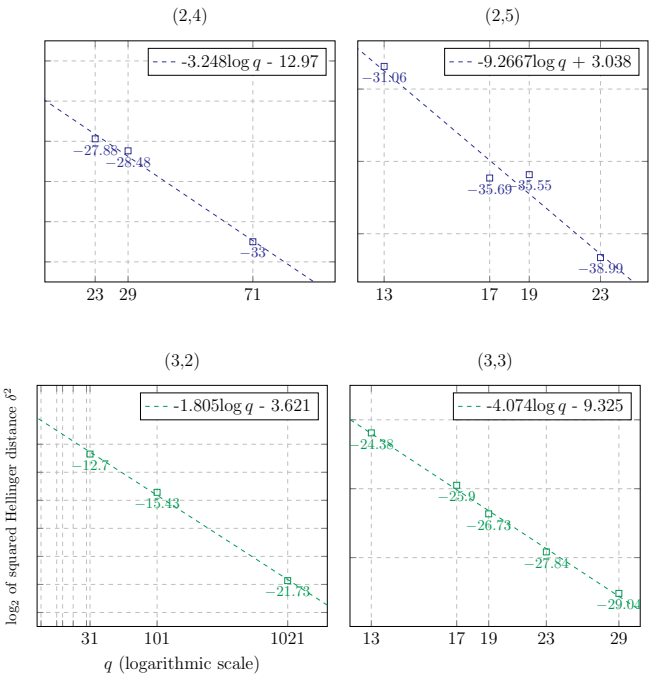

Figure 3.3: Empirical distance in pairwise intersection size over small fields with different choices of (m,κ). In each of the experiments, we sample $t = 1$M curves and estimate the pdf of the intersection size based on the number of pairwise intersections (of any given size) among all possible $\binom{t}{2}$ pairs. Dashed line denotes linear regression.

3.5 Implementation

We next discuss our implementation choices: our choice for the pseudorandom permutation used to permute the database encoding, and the pseudorandom function which we

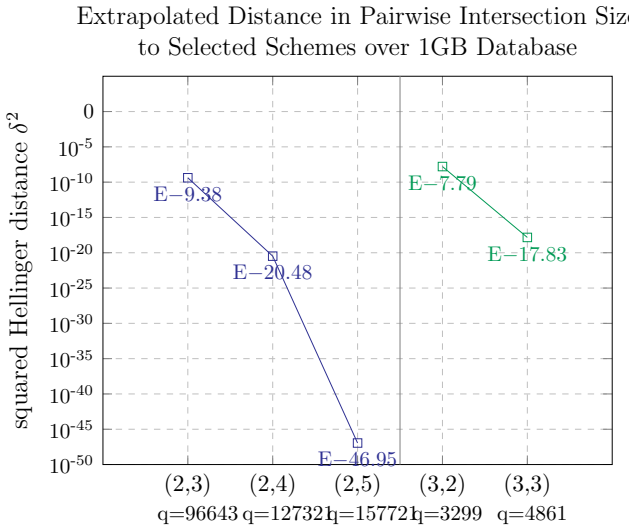

Figure 3.4: The extrapolated Hellinger distance in pairwise intersection size over the fields used in the instances from Tables 3.1 and 3.4, due to the linear regression over the results from Figure 3.3. The extrapolated distances are the basis for the security evaluation in Figure 3.1.

use for encrypting the database entries.

3.5.1 The Pseudo-Random Permutation

We implement the PRP based on the Golang **fpe** package for Format-Preserving Encryption (FPE), which provides an implementation of the NIST-approved FF1 mode of operation for FPE. We modify and extend it to match our requirements for permuting codewords in $\mathbb{F}_q^m$.

Parameter Selection. The aforementioned package contains an encryption algorithm for messages of length b, ranging from 2 to 128, and supports a radix r, ranging from 2 to 62. Consequently, it supports permutations of r^b integers, namely a maximal range of 62^{128}, which well exceeds our requirements. However, our total number of entries q^m might not match any pair of (r, b) values exactly. We choose a pair that minimizes the gap between r^b and q^m, ensuring r^b is the smallest value greater than q^m. We show that, despite the gap, we can still compute a permutation over q^m. Due to the flexibility of the

parameter r, we can achieve a permutation with 2 FPE operations for each position at average. We describe our adaptation in Algorithm 1 and analyze it in Claim 3.5.1.

Algorithm 1 PRP from FPE

Require: An FPE function E over $[r]^b$.
Ensure: A permutation P over $[n]$, for $n < r^b$.

```
1: procedure P(i ∈ [n])
2:     j ← E(i)
3:     while j > n do
4:         j ← E(j)
5:     end while
6:     return j
7: end procedure
```

Claim 3.5.1. *The function P computed in Algorithm 1 is a permutation over $[n]$, given that E is an FPE function E that operates on messages of length b with radix r. Further, the algorithm makes $\frac{r^b}{n}$ applications of E on average for a random input, and at most $r^b - n$ applications at any input.*

Proof. The function E defines a permutation over the range $[r^b]$. For any $i \in [r^b]$, there exists a unique cycle $C_i = (j_0 = i, j_1, ..., j_k = i)$, where $E(j_\ell) = j_{\ell-1}$ for all ℓ, and all elements in the cycle (except for first and last) are unique.

The output of Algorithm 1 on input i is the first element in this cycle, starting from j_1, that is within the range $[n]$. Since i itself is within this range, the algorithm is guaranteed to produce an output. Furthermore, since there are at most $r^b - n$ elements outside this range, the algorithm will find a solution within at most $r^b - n$ iterations. The average runtime follows from the assumption that the j_ℓ's are pseudorandom and, therefore, $j_\ell \in [n]$ is a Bernoulli with bias n/r^b.

We now show that P is injective over $[n]$. Assume that $P(i) = P(j) = k$ for some distinct $i, j \in [n]$. By the above observation, both C_i and C_j must contain k. This implies that C_i and C_j are, in fact, identical. In that cycle, k either resides on the path from i to j, in which case $P(j)$ cannot possibly be equal to k since i precedes it when starting

the walk from j, or k resides on the path between j and i, in which case a $P(i) \neq k$ by symmetry. □

3.5.2 AES for Data Encryption

Recall we use a pseudorandom function g to encrypt the database items in our sk-PIR scheme. We use AES to implement the PRF as $g_{\mathsf{sk}}(i) = \mathsf{AES}_{\mathsf{sk}}(i) \mod q$, for $i \in [N]$. Here, $\mathsf{AES}_{\mathsf{sk}}$ denotes AES encryption under the secret key sk. Using a 128-bit AES encryption, the output of the function g is a quasi-uniform with a bias of at most $\frac{1}{2^{128}}$ towards the first $2^{128} \mod q$ values, where the bias is defined as $\Delta = \max(|\frac{1}{q} - P(g(y) = i)|)$ for $i \in \{0, 1, ..., q-1\}$. Thus, we have

$$\Delta = \max_{b \in \{0,1\}} \left| \frac{1}{q} - \frac{\frac{2^{128} - (2^{128} \mod q)}{q} + b}{2^{128}} \right| < \frac{1}{2^{128}},$$

which is sufficiently small in our fine-grained setting.

3.6 Performance

We evaluate the performance of our base scheme via a series of micro-benchmarks with different values of (m,κ).

We measure the performance of our client on a MacBook Pro running macOS 14.5 (BuildVersion: 23F79). The machine is equipped with an Apple M3 Pro CPU, which has 11 cores, and 18 GB of RAM. We use the same benchmarking environment for all experiments and run our code using Go version 1.22.3 and Julia version 1.10.3. We use single-threaded execution for all experiments and report running times averaged over a minimum of 10 trials. The results are presented in Table 3.4. Our code is available at `https://anonymous.4open.science/r/SkipperFastSecretKeyPIR`.

We measure the server-side computation of our scheme on a Linux machine operating

Table 3.4: Online runtime per-query (in ms) for different values of m and κ. Time includes both client- and server-side computation. The value of q is calculated to match a total database-size of around 1GB.

κ	$m = 2, n = 2^{29}$ (Client, Server, q)	$m = 3, n = 2^{29.5}$ (Client, Server, q)	$m = 4, n = 2^{30}$ (Client, Server, q)
2	17.072, 4.88, 65537	1.031, 0.42, 3299	0.276, 0.075, 769
3	25.009, 10.08, 96643	1.602, 0.47, 4861	0.364, 0.11, 1129
4	34.789, 17.60, 127321	2.119, 0.62, 6389	0.524, 0.16, 1489
5	44.124, 21.71, 157721	2.433, 0.77, 7919	0.670, 0.21 1847

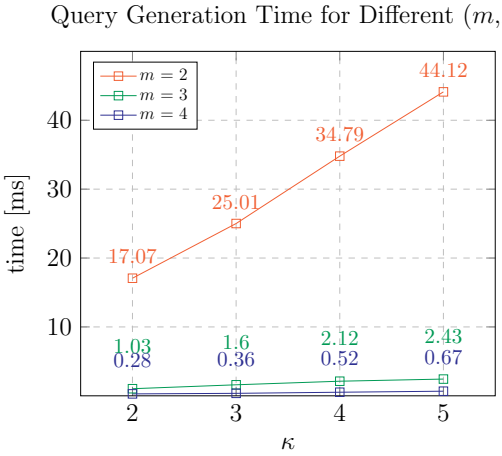

Figure 3.5: Query generation time for (m, κ) combinations. Value of q is chosen to ensure the database size is approximately 1GB.

under kernel version 5.15.0-69-generic on Ubuntu ($x86_64$ architecture). The system is powered by a 32-Core Processor running at 3.0 GHz and equipped with 125 GB of RAM. For a database with 2^{32} entries, where each entry is 16 bits, it takes $4.88ms$ to perform the computation for a single query, which constitutes of applying a linear function over 2^{16} entries. For the same database size yet different parametrization (corresponding to the second row in Table 3.1), a linear function over 2^{12} is evaluated by the server, and takes $0.4ms$ at average. We note that in such a setup, a scan over the full database takes 1.7 seconds at average. Due to the lightweight nature of the server-side computation in our construction, which can be easily parallelized and/or distributed, we estimate it to be

highly efficient and not to constitute the main computational workload for our scheme.

Comparison with Existing Protocols over 1GB. Most works in the PIR literature measure performance over a database of size 1 GB. We refer the reader to [HHCG+22], where a detailed comparison to various existing protocols is provided. In Table 3.1, we compare our scheme with selected competitive works over 1 GB databases. The choice of record size for each of the schemes matches the preferred size suggested in the respective work.

Offline Pre-processing. The pre-processing phase comprises two steps: a Reed-Muller (RM) encoding and the application of a pseudo-random permutation on the RM-encoded database. Besides an initial pre-processing, a client may want re-perform the database pre-processing to as a security measure that resets any advantage gained by an adversary. Note that the RM encoding has nothing to do with security and may be performed only once for any given database. Thus, the only step required in such a scenario is applying a new fresh pseudo-random permutation over the encoded database.

We ran the RM encoding for a database with 2^{29} entries, which results in an RM codeword of size 8G. We use the basic approach of Lagrange interpolation and compute using single thread. As a result, the encoding takes 23 hours. We stress, however, that the runtime of the encoding can be potentially boosted using FFT. Additionally, we benchmarked the runtime of permuting an RM-encoded database with 2^{32} entries, at 1357 seconds. In a scenario where the database is re-permuted once every $n = 2^{29}$ queries, such a runtime translates into an amortized $2.53\mu s$ per query.

Online Performance over Smaller Databases. We test the performance of our framework when instantiated over databases of size smaller than 1GB as well. We present the obtained benchmarks in Figure 3.6 and Figure 3.7.

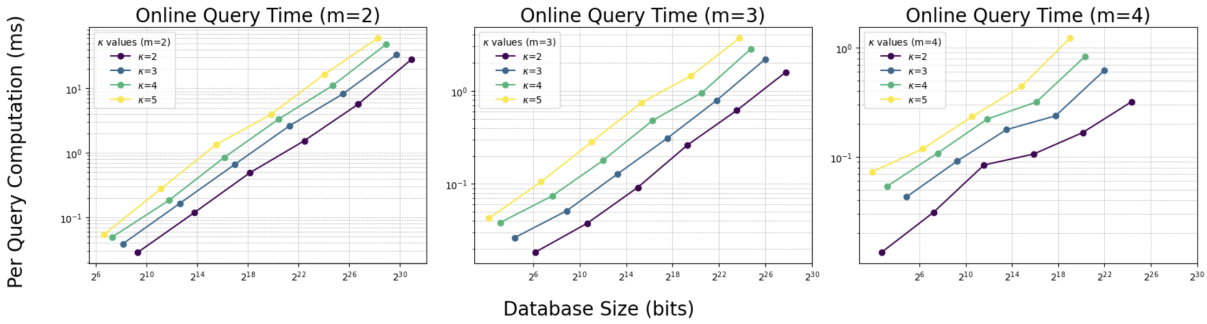

Figure 3.6: Online query time for different database size n, where each entry is a field element determined by the choice of (m, κ)

.

Figure 3.7: Client-to-Server Communication for different database size n, where each entry is a field element determined by the choice of (m, κ)

.

Bibliography

[ABG+24] Hilal Asi, Fabian Boemer, Nicholas Genise, Muhammad Haris Mughees, Tabitha Ogilvie, Rehan Rishi, Guy N. Rothblum, Kunal Talwar, Karl Tarbe, Ruiyu Zhu, and Marco Zuliani. Scalable private search with wally, 2024. 70

[ACLS18] Sebastian Angel, Hao Chen, Kim Laine, and Srinath Setty. Pir with compressed queries and amortized query processing. In *2018 IEEE symposium on security and privacy (SP)*, pages 962–979. IEEE, 2018. 64

[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. 25, 56, 57

[AG11a] 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. 10, 37

[AG11b] 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. 66, 88

[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. 4, 15

[AIK07] Benny Applebaum, Yuval Ishai, and Eyal Kushilevitz. Cryptography with constant input locality. In *Annual International Cryptology Conference*, pages 92–110. Springer, 2007. 21, 114

[AIK09] Benny Applebaum, Yuval Ishai, and Eyal Kushilevitz. Cryptography with constant input locality. *J. Cryptol.*, 22(4):429–469, 2009. 14, 32

[AIVG22] Kinan Dak Albab, Rawane Issa, Mayank Varia, and Kalman Graffi. Batched differentially private information retrieval. In *31st USENIX Security Symposium (USENIX Security 22)*, pages 3327–3344, 2022. 70

[Ajt10] Miklós Ajtai. Oblivious rams without cryptographic 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. 14

[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. 7, 30

[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. 15

[AMR25] Damiano Abram, Giulio Malavolta, and Lawrence Roy. Trapdoor hash functions and PIR from low-noise LPN. Cryptology ePrint Archive, Paper 2025/416, 2025. 7, 30

[AYA+21] Ishtiyaque Ahmad, Yuntian Yang, Divyakant Agrawal, Amr El Abbadi, and Trinabh Gupta. Addra: Metadata-private voice communication over fully untrusted infrastructure. In *15th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 21)*, 2021. 64

[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. 32

[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. 4, 15

[BCH+25] Fabrice Benhamouda, Caicai Chen, Shai Halevi, Yuval Ishai, Hugo Krawczyk, Tamer Mour, Tal Rabin, and Alon Rosen. Encrypted matrix-

vector products from secret dual codes. Cryptology ePrint Archive, Paper 2025/858, 2025. 15, 16

[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. 5, 6, 14, 29

[BHMW21] Elette Boyle, Justin Holmgren, Fermi Ma, and Mor Weiss. On the security of doubly efficient pir. *Cryptology ePrint Archive*, 2021. 2, 10, 14, 60, 73

[BHW19a] 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. 2, 10, 14

[BHW19b] Elette Boyle, Justin Holmgren, and Mor Weiss. Permuted puzzles and cryptographic hardness. In *Theory of Cryptography Conference*, pages 465–493. Springer, 2019. 60, 80

[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, 9

[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. 2, 4, 5, 9, 10, 11, 14, 15, 27, 36, 52, 60, 61, 71, 72, 73, 75, 77, 78, 79, 80, 81, 82, 86

[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. 7, 30

[BN25] Mark Braverman and Stephen Newman. Sublinear-overhead secure linear algebra on a dishonest server. *CoRR*, abs/2502.13060, 2025. 13, 24, 57, 58

[BRSV17] Marshall Ball, Alon Rosen, Manuel Sabin, and Prashant Nalini Vasudevan. Average-case fine-grained hardness. In Hamed Hatami, Pierre McKenzie, and Valerie King, editors, *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2017, Montreal, QC, Canada, June 19-23, 2017*, pages 483–496. ACM, 2017. 60

[BW21] Keller Blackwell and Mary Wootters. A note on the permuted puzzles toy conjecture. *arXiv preprint arXiv:2108.07885*, 2021. 2, 10, 14, 60

[CDH20] David Cash, Andrew Drucker, and Alexander Hoover. A lower bound for one-round oblivious ram. In *Theory of Cryptography: 18th International Conference, TCC 2020, Durham, NC, USA, November 16–19, 2020, Proceedings, Part I*, page 457–485, Berlin, Heidelberg, 2020. Springer-Verlag. 3

[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. 5, 13, 20, 31, 32, 41, 63, 87, 88

[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, 18

[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. 2, 5, 10, 11, 14, 15, 27, 36, 60, 61, 71, 72, 73, 75, 77, 78, 79, 81, 82, 86

[CIMR25] Caicai Chen, Yuval Ishai, Tamer Mour, and Alon Rosen. Secret-key PIR from random linear codes. Cryptology ePrint Archive, Paper 2025/646, 2025. 4

[CK20] Henry Corrigan-Gibbs and Dmitry Kogan. Private information retrieval with sublinear online time. In *Advances in Cryptology–EUROCRYPT 2020: 39th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, May 10–14, 2020, Proceedings, Part I 39*, pages 44–75. Springer, 2020. 3, 14

[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. 31

[CMS99] Christian Cachin, Silvio Micali, and Markus Stadler. Computationally private information retrieval with polylogarithmic communication. In Jacques Stern, editor, *Advances in Cryptology - EUROCRYPT '99, International Conference on the Theory and Application of Cryptographic Techniques, Prague, Czech Republic, May 2-6, 1999, Proceeding*, volume 1592 of *Lecture Notes in Computer Science*, pages 402–414. Springer, 1999. 24

[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. 15

[Dic58] Leonard Eugene Dickson. Linear groups, with an exposition of the galois field theory. 1958. 118

[DKL09] Yevgeniy Dodis, Yael Tauman Kalai, and Shachar Lovett. On cryptography with auxiliary input. In Michael Mitzenmacher, editor, *Proceedings of the 41st Annual ACM Symposium on Theory of Computing, STOC 2009, Bethesda, MD, USA, May 31 - June 2, 2009*, pages 621–630. ACM, 2009. 5, 6, 7, 8, 9, 13, 20, 31, 32, 41

[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. 14

[DMO00] Giovanni Di Crescenzo, Tal Malkin, and Rafail Ostrovsky. Single database private information retrieval implies oblivious transfer. In Bart Preneel, ed-

itor, *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. 4

[Döt14] Nico Döttling. *Cryptography based on the Hardness of Decoding*. PhD thesis, Karlsruhe Institute of Technology, 2014. 21, 42

[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. 42

[DVV16] Akshay Degwekar, Vinod Vaikuntanathan, and Prashant Nalini Vasudevan. Fine-grained cryptography. In Matthew Robshaw and Jonathan Katz, editors, *Advances in Cryptology - CRYPTO 2016 - 36th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2016, Proceedings, Part III*, volume 9816 of *Lecture Notes in Computer Science*, pages 533–562. Springer, 2016. 60

[FLLP24] Ben Fisch, Arthur Lazzaretti, Zeyu Liu, and Charalampos Papamanthou. Single server PIR via homomorphic thorp shuffles. *Cryptology ePrint Archive*, 2024. 64

[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. 15

[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. 31

[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. 15

[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. 114

[GO96a] Oded Goldreich and Rafail Ostrovsky. Software protection and simulation on oblivious rams. *J. ACM*, 43(3):431–473, May 1996. 3

[GO96b] Oded Goldreich and Rafail Ostrovsky. Software protection and simulation on oblivious rams. *J. ACM*, 43(3):431–473, May 1996. 14

[Gol87] O. Goldreich. Towards a theory of software protection and simulation by oblivious rams. In *Proceedings of the Nineteenth Annual ACM Symposium on Theory of Computing*, STOC '87, page 182–194, New York, NY, USA, 1987. Association for Computing Machinery. 3

[Gol01] Oded Goldreich. *Foundations of Cryptography: Volume 1, Basic Tools*. Cambridge University Press, Cambridge, UK, 2001. 27

[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. 12

[HHCG+22] 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. Cryptology ePrint Archive, Paper 2022/949, 2022. https://eprint.iacr.org/2022/949. 64, 95

[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. 52

[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. 12

[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. 4

[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. 4, 15

[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. 18

[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. 14

[KCG21] Dmitry Kogan and Henry Corrigan-Gibbs. Private blocklist lookups with checklist. In *30th USENIX Security Symposium (USENIX Security 21)*, pages 875–892, 2021. 3

[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. 36

[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, 12, 18, 24

[Lip05] Helger Lipmaa. An oblivious transfer protocol with log-squared communication. In Jianying Zhou, Javier López, Robert H. Deng, and Feng Bao, editors, *Information Security, 8th International Conference, ISC 2005, Sin-*

gapore, September 20-23, 2005, Proceedings, volume 3650 of *Lecture Notes in Computer Science*, pages 314–328. Springer, 2005. 24

[LMRSW23] Baiyu Li, Daniele Micciancio, Mariana Raykova, and Mark Schultz-Wu. Hintless single-server private information retrieval. Cryptology ePrint Archive, Paper 2023/1733, 2023. `https://eprint.iacr.org/2023/1733`. 64, 65

[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. 2, 6, 12, 15

[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. 15

[LP24] Arthur Lazzaretti and Charalampos Papamanthou. Single pass client-preprocessing private information retrieval. Cryptology ePrint Archive, Paper 2024/303, 2024. `https://eprint.iacr.org/2024/303`. 3

[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. 2

[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. 10

[MBFK16] Carlos Aguilar Melchor, Joris Barrier, Laurent Fousse, and Marc-Olivier Killijian. Xpir: Private information retrieval for everyone. *Proceedings on Privacy Enhancing Technologies*, pages 155–174, 2016. 64

[MCR21] Muhammad Haris Mughees, Hao Chen, and Ling Ren. Onionpir: Response efficient single-server pir. In *Proceedings of the 2021 ACM SIGSAC conference on computer and communications security*, pages 2292–2306, 2021. 64, 65

[Mer78] Ralph C. Merkle. Secure communications over insecure channels. *Commun. ACM*, 21(4):294–299, 1978. 60

[MIR23] Muhammad Haris Mughees, Sun I, and Ling Ren. Simple and practical amortized sublinear private information retrieval. Cryptology ePrint Archive, Paper 2023/1072, 2023. `https://eprint.iacr.org/2023/1072`. 3

[MSR23] Muhammad Haris Mughees, I Sun, and Ling Ren. Simple and practical amortized sublinear private information retrieval. *Cryptology ePrint Archive*, 2023. 64

[MW22] Samir Jordan Menon and David J Wu. Spiral: Fast, high-rate single-server pir via fhe composition. In *2022 IEEE Symposium on Security and Privacy (SP)*, pages 930–947. IEEE, 2022. 64, 65

[MW24] Samir Jordan Menon and David J Wu. Ypir: High-throughput single-server pir with silent preprocessing. *Cryptology ePrint Archive*, 2024. 64

[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. 24

[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. 15

[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. 31

[SS92] Vladimir Michilovich Sidel'nikov and Sergey O Shestakov. On an encoding system constructed on the basis of generalized reed–solomon codes. *Diskretnaya Matematika*, 4(3):57–63, 1992. 129

[Ste98] Julien P. Stern. A new efficient all-or-nothing disclosure of secrets protocol. In Kazuo Ohta and Dingyi Pei, editors, *Advances in Cryptology - ASIACRYPT '98, International Conference on the Theory and Applications of Cryptology and Information Security, Beijing, China, October 18-22, 1998,*

Proceedings, volume 1514 of *Lecture Notes in Computer Science*, pages 357–371. Springer, 1998. 24

[SW13] Christopher J. Shallue and Ian M. Wanless. Permutation polynomials and orthomorphism polynomials of degree six. *Finite Fields Their Appl.*, 20:84–92, 2013. 118

[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. 14

[TDG16] Raphael Toledo, George Danezis, and Ian Goldberg. Lower-cost epsilon-private information retrieval. *Proceedings on Privacy Enhancing Technologies*, 2016, 04 2016. 70

[Vid03] Rene Esteban Vidal. *Generalized principal component analysis (gpca): an algebraic geometric approach to subspace clustering and motion segmentation.* PhD thesis, 2003. AAI3121739. 31

[VZ25] Vinod Vaikuntanathan and Or Zamir. Improving algorithmic efficiency using cryptography. *CoRR*, abs/2502.13065, 2025. 13, 24, 57, 58

[YZ21] Yu Yu and Jiang Zhang. Smoothing out binary linear codes and worst-case sub-exponential hardness for LPN. In Tal Malkin and Chris Peikert, editors, *Advances in Cryptology – CRYPTO 2021, Part III*, volume 12827 of *Lecture Notes in Computer Science*, pages 473–501, Virtual Event, August 16–20, 2021. Springer, Cham, Switzerland. 13

[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. 7, 30

[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. 14

[ZPSZ23] Mingxun Zhou, Andrew Park, Elaine Shi, and Wenting Zheng. Piano: Extremely simple, single-server pir with sublinear server computation. *Cryptology ePrint Archive*, 2023. 3, 64, 65

Appendix A: Proofs for Chapter 2

.1 Proof of Lemma 2.4.2

Lemma 2.4.2 (From LPN to Matrix-LPN)**.** (k, ε)-LPN *implies* (k, n, ε)-Matrix-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}. □

.2 Proof of Lemma 2.4.1

Lemma 2.4.1 (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 $\mathbf{A}' = \mathbf{A} - \mathbf{zr}^\intercal$, and return its output.

The hardcore-bit is computed correctly with probability $1/2 + \mathrm{poly}(\lambda)$ since it holds that $\mathbf{v} = \mathbf{A}'\mathbf{s} + \mathbf{zr}^\intercal\mathbf{s} + \mathbf{e}$. 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 $1/\mathrm{poly}(\lambda)$ 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. □

Appendix B: Further Cryptanalysis for Chapter 3

In this section, we compliment our heuristics regarding the security of the augmented schemes against statistical attacks with analysis and experiment. In our cryptanalysis we consider the base scheme. Nonetheless, most attacks (and huerstics regarding the lack thereof) can be generalized to capture the augmented schemes by considering an adversary that collects sufficiently more statistics.

.3 Preliminary Statistical Analysis

Whenever $m, \kappa \in \mathbb{N}$ are implicit by context, we let $\mathcal{C}$ denote the set of all curves over $\mathbb{F}^m$ of degree at most κ. For any $\alpha \in \mathbb{F}^m$, we denote by $\mathcal{C}_\alpha$ the set of any such curve C satisfying $C(0) = \alpha$. Additionally, for any curve C, we let

$$V(C) = \langle C(t) \rangle_{t \in \mathbb{F}^*}, \tag{1}$$

and denote

$$\mathcal{V} = \{V(C) \mid C \in \mathcal{C}\} \quad \text{and} \quad \mathcal{V}_\alpha = \{V(C) \mid C \in \mathcal{C}_\alpha\}.$$

In particular, a subset $V \leftarrow \mathcal{V}_\alpha$, that undergoes re-labeling under the permutation π, distributes like a PIR query corresponding to the database location associated with $\boldsymbol{\alpha}$.

Proposition .3.1. *For any $\kappa, m \in \mathbb{N}$ and $\alpha \in \mathbb{F}^m$, $|\mathcal{V}_\alpha| = |\mathcal{V}|/q^m$.*

Proof. The proposition follows from the fact that $|\mathcal{V}_\alpha|$ is equal over all $\alpha \in \mathbb{F}^m$ due to symmetry and that for any α, α', the sets $\mathcal{V}_\alpha$ and $\mathcal{V}_{\alpha'}$ are disjoint (the latter follows from the correctness of the scheme). □

In the following proposition, we give an upper bound on the cardinality of $\mathcal{V}$ and, therefore, on the distributions Random and Planted.

Proposition .3.2. *For any $\kappa, m \in \mathbb{N}$, it holds that*

$$|\mathcal{V}| \leq q^{m(\kappa+1)-1}/(q-1).$$

Proof. A straight-forward upper bound on the size of $\mathcal{V}$ is derived by counting the number of possible choices of m polynomials of degree at most κ, which bounds the number of possible curves in $\mathcal{C}$ (see Definition 3.2.2). There are $q^{m(\kappa+1)}$ such choices. Observe, however, that different choices of polynomials (i.e. curve) may produce equal value sets. For instance, any two curves C and C', where C is defined by polynomials $p_1, \ldots, p_m$ and C' by $p'_1, \ldots, p'_m$, where $p'_i = p_i \circ \varphi$, for a non-constant affine function φ over $\mathbb{F}$ (which is in particular a permutation) share the same value set. Thus, a tighter upper bound may be derived by dividing all choices of curves by the number of affine permutations over $\mathbb{F}$, which is $q(q-1)$. □

To argue security of our scheme against generic attack strategies, e.g. a birthday attack, it is crucial we give also a lower bound on the cardinality of $\mathcal{V}$. While the above upper bound is not tight, we conjecture it gives us a sufficiently good estimate, at least asymptotically.

Conjecture .3.1. *For any $\kappa, m \in \mathbb{N}$, it holds that*

$$|\mathcal{V}| \to q^{m(\kappa+1)-2},$$

when $q \to \infty$.

Looking back at the proof of Proposition .3.2, observe that, while we account for "collisions" in the value set obtained by affine permutations, we do not consider collisions

incurred by permutations of higher degree over $\mathbb{F}$. While we are not aware of any complete rigorous analysis for the number of low-degree polynomials, we have strong evidence that it is insignificant. Dickson [Dic58] enumerates all such polynomials of degree less or equal to 5 using Hermite's criterion. He finds that the number of such polynomials of normalized form is constant, for *any* field size q. Shallue and Wanless [SW13] reach a similar conclusion for degree 6.

Additionally, note that even non-bijective maps may possibly result in such collisions between two distinct curves. Any such map, however, is associated with collisions involving polynomials satisfying a specific structure in their value sets. Hence, roughly speaking, such collisions constitute an insignificant fraction of all polynomials.

We support our heuristics above, on which we draw Conjecture .3.1, by empirical data as demonstrated in Fig. 8.

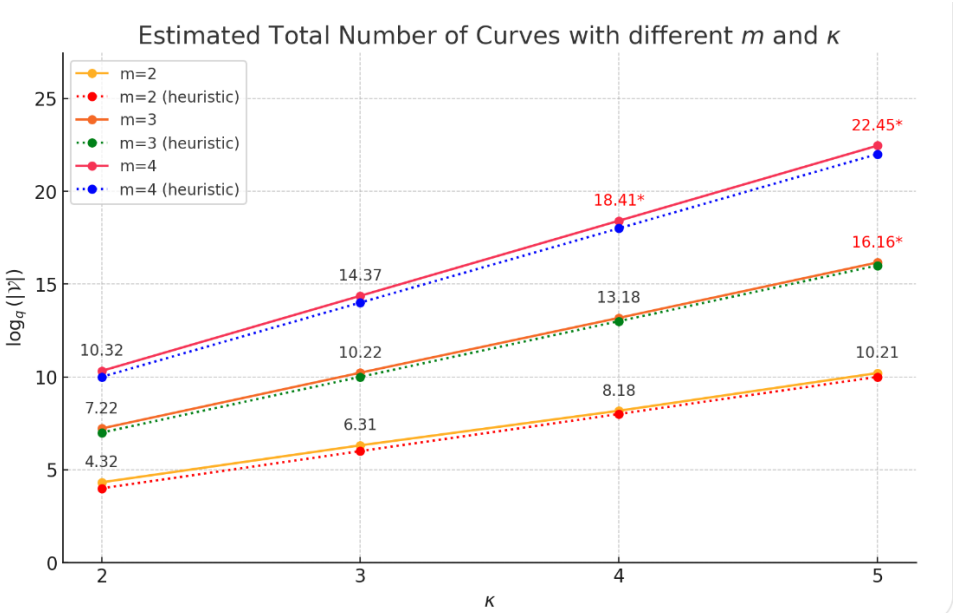

Figure 8: Estimated total number of curves for different values of m and κ. The experiments were conducted to mimic a birthday attack, i.e. to find a first repetition in random samples. By the birthday paradox, the total number of curves is estimated to be the square of number of samples required for a repetition. Each point represents an average of 1000 runs, and points marked with an asterisk (*) are extrapolated.

.4 A Birthday Attack

An immediate consequence of Propositions .3.1 and .3.2 is a birthday-attack solver for the low-degree planted point problem, which makes $\sqrt{|\mathcal{V}_\alpha|} = q^{m\kappa/2-1}$ queries and has constant advantage. More generally, one may obtain the following trade-off between the number of queries and advantage.

Proposition .4.1 (Birthday Attack)**.** *For any $\alpha \in (0,1)$, there exists against our constructions (and, in particular, the PLDN problem) that requires observing $\alpha \cdot q^{m\kappa/2-1}$ (non-dummy) queries and has advantage $1 - \Theta(e^{-\alpha^2})$.*

Assuming Conjecture .3.1, Proposition .4.1 captures the best trade-off that can be achieved by a birthday attack.

.5 A Statistical Attack on The Degree-1 Case

Next, we show that any instantiation of our framework with $\kappa = 1$ yields an insecure scheme. In particular, against the base scheme of such a setting, there exists a distinguisher that can distinguish between two queries corresponding to the same database item and two queries corresponding to different database items. We stress that, following the same outline, the PLDN problem is also easy to solve when $\kappa = 1$.

Lemma .5.1. *Assume Conjecture .3.1 holds*[6]*. There exists an algorithm that breaks the security of the base scheme when $\kappa = 1$, with advantage $\Omega(1/q^{m-1})$ in time and space linear in q, by observing only two client queries.*

The attack exploits the fact that two distinct degree-1 curves cannot possibly intersect at more than a single point. Thus, two distinct samples from $\mathsf{PLDN}(\alpha)$, for any $\alpha \in \mathbb{F}^m$,

[6]Even much looser forms of the conjecture imply attacks with sufficiently good advantage.

never intersect, whereas two samples corresponding to two uniformly random values of α do with good probability.

We define the attack algorithm $\mathcal{A}$ as follows.

$\mathcal{A}^{(\cdot)}(1^q)$:

1. Observe two PIR queries to obtain value sets V_1, V_2.
2. If $0 < |V_1 \cap V_2| < q$, output $b = 1$.
3. Otherwise, output $b = 0$.

To prove $\mathcal{A}$'s advantage, we start by bounding the intersection of any two degree-1 curves.

Claim .5.1. *For any pair of degree-1 curves C_1, C_2 over $\mathbb{F}^m$ (recall Definition 3.2.2), it holds that either $C_1 = C_2$ or $|C_1 \cap C_2| < 2$.*

Proof. Assume $\boldsymbol{\alpha}_1$ and $\boldsymbol{\alpha}_2$ are two distinct points on degree-1 curve C, then the rest of the points on the curve are an affine combination of those 2 points which has the form: $\delta\boldsymbol{\alpha}_1 + (1-\delta)\boldsymbol{\alpha}_2$, for all $\delta \in \mathbb{F}$.

So if two degree-1 curves coincide on two points, they must coincide on all other points. □

By the above claim, for any $\alpha \in \mathbb{F}$,

$$\Pr_{V_1, V_2 \leftarrow \mathsf{PLDN}^{\pi}(\alpha)}[0 < |V_1 \cap V_2| < q] = 0. \tag{2}$$

It remains to lower bound the probability of $\mathcal{A}$ outputting 1 in the case where V_1 and V_2 are sampled from PLDN^{π} for uniform values of $\boldsymbol{\alpha}$, namely from $\mathcal{V}$. Let C_1, C_2 be two uniformly random degree-1 curves over $\mathbb{F}^m$. Due to Claim .5.1, we may calculate

the probability that $V_1 = V(C_1)$ and $V_2 = V(C_2)$ intersect at one point as follows (recall definition in Eq. (1))

$$\begin{aligned}
&\Pr_{V_1,V_2}[|V_1 \cap V_2| = 1] \\
&= \sum_{t \in \mathbb{F}\setminus\{0\}} \Pr_{V_1,C_2}[C_2(t) \in V_1] - \Pr[V_1 = V_2] \\
&\geq (q-1)/q^m - 1/|\mathcal{V}| = \Omega(1/q^{m-1}),
\end{aligned}$$

where the last equality is due to Conjecture .3.1. This, together with Eq. (2), completes the proof of Lemma .5.1.

.6 Resilience against 1-Sample Statistics

We consider attacks that evaluate *any* statistic over individual samples, ignoring any potential inter-dependencies. We show that such attacks are bound to fail since the distribution of a single sample from $\mathsf{PLDN}^{\pi}(\boldsymbol{\alpha})$ is independent in $\boldsymbol{\alpha}$.

Proposition .6.1. *For any $\alpha, \beta \in \mathbb{F}^m$, a single sample from $\mathsf{PLDN}^{\pi}(\boldsymbol{\alpha})$ distributes identically to a single sample from $\mathsf{PLDN}^{\pi}(\beta)$.*

Proof. Since π is a uniformly random permutation, it suffices to argue equivalence of the multiplicity vectors corresponding to the two samples (namely, the multiplicities of distinct elements in the sample, which is oblivious in the permutation π). The equivalence then follows from the fact that, for any $\boldsymbol{\alpha}, \beta \in \mathbb{F}^m$, there exists a 1-1 correspondence between the image of $\mathsf{PLDN}(\boldsymbol{\alpha})$ and $\mathsf{PLDN}(\beta)$ (assuming π is identity), which maps any curve $C \in \mathsf{PLDN}(\boldsymbol{\alpha})$ to the curve $C + \beta - \alpha$, which has the same multiplicity vector. □

.7 Resilience against Intersection-Size Statistics

Next, we investigate attacks that look at the *size of intersections* between multiple samples from the oracle. In Section 3.4.2, we provide empirical evidence that the squared Hellinger distance in intersection sizes, between pairs of curves corresponding to the same client request and pairs of random curves, is small enough in our schemes to support our conjectured levels of security.

In what comes next, we support these findings with rigorous analysis which, albeit very partial, we believe sheds light on the source of this behavior and on the (im)plausibility of more general statistical attacks.

Ideally, to rigorously prove that intersection-size statistics are useless to distinguish between, say the distributions $\mathcal{V}_{\boldsymbol{\alpha}}$ and $\mathcal{V}$ (recall their correspondence with the distribution of queries in our base scheme), we would like to show that the distribution of the intersection size between random $r > 1$ samples is (statistically) indistinguishable under the two distributions. In particular, recall that such samples consist of multisets over $\mathbb{F}^m$ and, therefore, intersections thereof are multisets as well. Our formal analysis, however, falls short of this goal. Instead, we are able to show indistinguishability holds under a related, yet restricted, statistic, which is formalized in Lemma .7.1.

An intuitive way to view our rigorous result is as follows: If the probability for an intersection of size ℓ between two samples is close under the two distributions, then such overlaps have a similar structure (recall the samples consist of multi-sets and might therefore induce many possible intersecting combinations). Also the opposite holds; if intersections of size ℓ look the same under the two distributions, then the probability of them occurring is close. We believe our proof of this weaker statement is insightful and supports our heuristic that such an indistinguishability holds against stronger statistics concerning the overlap between queries.

We stress that our analysis below extends to t-wise intersections for $t > 2$, namely

overlaps shared by t samples, where the proven notion of statistical distance decreases exponentially in t.

The following proposition follows immediately from the $(\kappa+1)$-independence of points on a random degree-κ curve.

Proposition .7.1. *Fix $t_1, \ldots, t_\kappa \in \mathbb{F}^*$, and let $E(C)$ be a function over curves that depends solely on the values $C(t_1), \ldots, C(t_\kappa)$. Then, the distribution of $E(R)$ for $R \leftarrow \mathcal{C}$ is equivalent to the distribution of $E(P)$ for $P \leftarrow \mathcal{C}_{\boldsymbol{\alpha}}$, for any fixed $\boldsymbol{\alpha} \in \mathbb{F}^m$.*

For any two lists V_1, V_2 (e.g. samples form $\mathcal{V}$ or $\mathcal{V}_{\boldsymbol{\alpha}}$),let $I(V_1, V_2)$ be the number of distinct values in $V_1 \cap V_2$ (i.e. the size of the intersection when reducing V_1 and V_2 to their set representation). We denote by $I_\ell(V_1, V_2)$ the predicate that takes 1 if and only if $I(V_1, V_2) = \ell$. Lastly, we denote by $\mu(V_1, V_2)$ the number of different maximal overlaps between V_1 and V_2 that contain distinct values. Formally, $\mu(V_1, V_2)$ is the number of all pairs of subsets $S_1 \subseteq V_1, S_2 \subseteq V_2$ of size $I(V_1, V_2)$ such that $I(S_1, S_2) = I(V_1, V_2)$.

Lemma .7.1. *Let $\boldsymbol{\alpha} \in \mathbb{F}^m$. Let $R_1, R_2 \leftarrow \mathcal{V}$ and $P_1, P_2 \leftarrow \mathcal{V}_{\boldsymbol{\alpha}}$. Then, for any $\ell > 0$, it holds that*

$$\Big| \mathbb{E}[I_\ell(R_1, R_2) \cdot \mu(R_1, R_2)] - \mathbb{E}[I_\ell(P_1, P_2) \cdot \mu(P_1, P_2)] \Big| \leq 2 \left(\frac{e\kappa}{q^{m-2}} \right)^\kappa .$$

Proof of Lemma .7.1

We start by rewriting

$$\begin{aligned}
&\mathbb{E}[I_\ell(R_1,R_2)\cdot\mu(R_1,R_2)] \\
&= \frac{1}{|\mathcal{C}|^2}\cdot \sum_{\substack{C_1,C_2\in\mathcal{C}:\\ I(V(C_1),V(C_2))=\ell}} \mu(V(C_1),V(C_2)) \\
&= \frac{1}{|\mathcal{C}|^2}\cdot \underbrace{\sum_{\substack{C_1,C_2\in\mathcal{C}:\\ I(V(C_1),V(C_2))=\ell}} \frac{1}{\ell!}\cdot \sum_{T,T'\in\mathbb{F}^\ell} 1}_{\mu},
\end{aligned} \tag{3}$$

where the second sum goes over any two lists $T=(t_1,\ldots,t_\ell)\in\mathbb{F}^\ell$ and $T'=(t'_1,\ldots,t'_\ell)\in\mathbb{F}^\ell$ of distinct field elements, such that $C(t_i)=C'(t'_i)=\gamma_i$ for all i, and $\{\gamma_i\}$ are all distinct. Note the $1/\ell!$ factor is due to the duplicity in the order of elements in T (which in particular determines the order over T').

Now, taking a reverse perspective, for any two lists $T=(t_1,\ldots,t_s)\in\mathbb{F}^s$ and $T'=(t'_1,\ldots,t'_s)\in\mathbb{F}^s$ of distinct field elements, we denote by $N(T,T')$ the number of low-degree curve pairs $C,C'\in\mathcal{C}$ such that $C(t_i)=C'(t'_i)=\gamma_i$ for all i, and $\{\gamma_i\}$ are all distinct. In words, $N(T,T')$ counts all pairs of curves that overlap at locations T and, resp., T', with distinct values. Then, swapping the summations in Eq. (3), we may bound

$$\mu \le \frac{1}{\ell!}\cdot \sum_{T,T'\in\mathbb{F}^\ell} N(T,T'), \tag{4}$$

where the sum goes over all lists T and T' of distinct field elements. The above is merely an upper bound and, in fact, not a very tight one, since we are also counting curves that overlap by more than ℓ distinct values. From this point, we proceed using the principle of inclusion-exclusion.

Consider pairs of curves that have overlaps containing $\ell+1$ distinct values. Any overlap between such a pair contains, in particular, $\ell+1=\binom{\ell+1}{\ell}$ distinct subsets of size

ℓ. Thus, we have counted such pairs, in the sum in Eq. (4), $\ell+1$ times for each different overlap it induces. Therefore, a second step is to exclude all $(\ell+1)$-overlaps from the sum to obtain the following lower bound

$$\mu \geq \frac{1}{\ell!} \sum_{T,T' \in \mathbb{F}^{\ell}} N(T,T') - \frac{1}{(\ell+1)!} \binom{\ell+1}{\ell} \cdot \sum_{T,T' \in \mathbb{F}^{\ell+1}} N(T,T').$$

Note that the above is a lower bound since, while we have accurately counted overlaps containing exactly ℓ or $\ell+1$ distinct values (once and "zero times", respectively), any pair that intersects in $\ell' > \ell+1$ points shows $\binom{\ell'}{\ell} - \binom{\ell+1}{\ell} \cdot \binom{\ell'}{\ell+1} < 0$ times in the sum above.

Iteratively recursing with the same logic, we obtain for any odd $0 < t < q-\ell$,

$$\sum_{i=0}^{t} \frac{(-1)^i}{(\ell+i)!} \cdot \alpha_i \cdot \sum_{T,T' \in \mathbb{F}^{\ell+i}} N(T,T') \leq \mu \leq \sum_{i=0}^{t+1} \frac{(-1)^i}{(\ell+i)!} \cdot \alpha_i \cdot \sum_{T,T' \in \mathbb{F}^{\ell+i}} N(T,T'),$$

where $\alpha_0 = 1$ and

$$\alpha_i = \sum_{j=0}^{i-1} \binom{\ell+i}{\ell+j} \cdot (-1)^{i-j-1} \cdot \alpha_j.$$

Hence, letting $p(T,T') = N(T,T')/|\mathcal{C}|^2$ denote the probability that random $C_1, C_2 \leftarrow \mathcal{C}$ satisfy $C_1(T) = C_2(T')$ with all-distinct evaluations (in other words, that C_1, C_2 where

counted in $N(T,T')$), we have that

$$\begin{aligned}
&\sum_{i=0}^{t} \frac{(-1)^i}{(\ell+i)!} \cdot \alpha_i \cdot \sum_{T,T' \in \mathbb{F}^{\ell+i}} p(T,T') \\
&\quad \leq \mathbb{E}[I_\ell(R_1,R_2) \cdot \mu(R_1,R_2)] \\
&\quad\quad \leq \frac{(-1)^i}{(\ell+i)!} \cdot \alpha_i \cdot \sum_{T,T' \in \mathbb{F}^{\ell+i}} p(T,T'). \qquad (5)
\end{aligned}$$

Identical bounds may be derived for the expectation over samples from the planted distribution, namely $P_1, P_2 \leftarrow \mathcal{V}_{\boldsymbol{\alpha}}$, when accordingly replacing any term $p(T,T')$ with the corresponding $p_{\boldsymbol{\alpha}}(T,T')$, which denotes the probability that random $P_1, P_2 \leftarrow \mathcal{V}_{\boldsymbol{\alpha}}$ is counted in $N(T,T')$. Consequently, and since $p(T,T') = p_{\boldsymbol{\alpha}}(T,T')$ for any T,T' of length at most κ (due to Proposition .7.1), Eq. (5) implies

$$\begin{aligned}
&\Big|\ \mathbb{E}[I_\ell(R_1,R_2) \cdot \mu(R_1,R_2)] - \mathbb{E}[I_\ell(P_1,P_2) \cdot \mu(P_1,P_2)]\ \Big| \\
&\leq \frac{2\alpha_{\kappa-\ell}}{\kappa!} \cdot \sum_{T,T' \in \mathbb{F}^{\kappa}} p(T,T') \leq 2\alpha_{\kappa-\ell} \cdot \kappa! \binom{q}{\kappa}^2 \cdot \frac{1}{q^{\kappa m}}, \qquad (6)
\end{aligned}$$

where the last inequality is due to the following proposition.

Proposition .7.2. *For any two fixed lists of distinct field elements $T = (t_1, \ldots, t_\kappa)$ and $T' = (t'_1, \ldots, t'_\kappa)$, it holds that $p(T,T') \leq 1/q^{\kappa m}$.*

Proof. Recall that $p(T,T')$ is the probability for two random curves $C_1, C_2 \leftarrow \mathcal{C}$ to satisfy: (1) $\forall i$, $C_1(t_i) = C_2(t'_i) = \gamma_i$ for some γ_i, and (2) all γ_i are distinct. In particular, $p(T,T')$ is upper-bounded by the probability that a random degree-κ curve evaluates to an apriori fixed set of values at $t_1, \ldots, t_\kappa$. By the $(\kappa+1)$-independence of the evaluations of a random degree-κ curve, this is $1/q^{\kappa m}$.

□

It remains, given Eq. (6), to bound the value of $\alpha_{\kappa-\ell}$. We finish the proof of the lemma via the following claim, a proof to which may be found in the full version.

Claim .7.1. *For any $0 \leq i \leq q - \ell$, it holds that $\alpha_i \leq (\ell + i)! \cdot e^i$.*

Proof. We first argue that

$$\alpha_i = \sum_{k=1}^{i} (-1)^{i-k} \sum_{\substack{s_1,\ldots,s_k \in \mathbb{N}: \\ s_1+\cdots+s_k=i}} \beta^{[i]}_{s_1,\ldots,s_k}$$

where

$$\beta^{[i]}_{s_1,\ldots,s_k} = \binom{\ell+i}{s_1}\binom{\ell+i-s_1}{s_2}\cdots\binom{\ell+i-\sum_{r=1}^{k-1} s_r}{s_k}.$$

The equality is immediate for $i = 0$ (recall $\alpha_0 = 1$). For $i > 0$, we prove by induction. Recall that α_i is the sum of terms of the form

$$(-1)^{i-j-1} \cdot \binom{\ell+i}{\ell+j} \cdot \alpha_j = (-1)^{i-j-1} \cdot \binom{\ell+i}{i-j} \cdot \alpha_j,$$

for $j = 0, \ldots, i-1$. Assume by induction that any such α_j is a sum consisting of a $\beta^{[j]}_{s_1,\ldots,s_k}$ term, with the corresponding phase $(-1)^{j-k}$, for any $s_1, \ldots, s_k$ that is a partitioning of the range $\{1, \ldots, j\}$. Then, evidently, and ignoring powers of (-1) for simplicity, the sum $\sum \binom{\ell+j}{i-j} \cdot \alpha_j$ consists precisely of such $\beta^{[i]}$ terms for all possible partitions of the range $\{1, \ldots, i\}$; any such partition is obtained by first determining $s_1 = i - j > 0$ then arbitrarily partitioning the range $\{1, \ldots, j\}$. Formally,

$$\begin{aligned}
\alpha_i &= \sum_{s_1=1}^{i} (-1)^{s_1-1} \binom{\ell+i}{s_1} \\
&\quad \times \sum_{\substack{s_2,\ldots,s_k \in \mathbb{N}: \\ s_2+\cdots+s_k=i-s_1}} (-1)^{(i-s_1)-(k-1)} \beta^{[i-s_1]}_{s_2,\ldots,s_k} \\
&= \sum_{\substack{s_1,\ldots,s_k \in \mathbb{N}: \\ s_1+\cdots+s_k=i}} (-1)^{i-k} \beta^{[i]}_{s_1,\ldots,s_k}.
\end{aligned}$$

Lastly, notice that $\beta^{[j]}_{s_1,\ldots,s_k}$ is, in fact, a telescopic product which reduces to

$$\beta^{[i]}_{s_1,\ldots,s_k} = \frac{(\ell+i)!}{s_1!s_2!\ldots s_k!}$$

and, therefore,

$$\begin{aligned}\alpha_i &= (\ell+i)! \cdot \sum_{k=1}^{i} (-1)^{i-k} \sum_{\substack{s_1,\ldots,s_k \in \mathbb{N}: \\ s_1+\cdots+s_k=i}} \frac{1}{s_1!s_2!\ldots s_k!} \\ &\leq (\ell+i)! \cdot \sum_{1 \leq s_1,\ldots,s_k \leq i} \frac{1}{s_1!s_2!\ldots s_k!}.\end{aligned}$$

The claim then follows since $\sum_{s=1}^{i} 1/s! < e$ for $i \in \mathbb{N}$. □

The proof of Lemma .7.1 is then complete by plugging in Claim .7.1 in Eq. (6).

We can leverage Intersection-Size Statistics to form an attack that clusters queries by index, based on the observation that an intersection of size κ^m can only occur between queries with different indices. This clustering method relies on the following property and is supported by an intuitive proof.

Proposition .7.3. *For a Stochastic Block Model with n blocks, where the within-block connection probability is zero and the cross-block connection probability is $p < \frac{1}{2\log(n)}$, an attack can identify the blocks after N samples, where $n \leq \frac{N}{2\log(N/p)}$.*

Proof. Construct a graph G where each vertex represents a sample, and two vertices are connected if they have intersection size k^m.

Clearly, G forms a Stochastic Block Model with zero probability of connections within blocks and cross-block connection probability p.

When n is large and p is small, G is sparse and approximates an ErdHos-Rényi graph $G(N, p \cdot \frac{n-1}{n})$, with an asymptotic chromatic number of approximately $\frac{N}{2\log(N/p)}$.

Since G is always n-colorable, if N satisfies $\frac{N}{2\log(N/p)} > n$, the only feasible n-coloring solution is the exact clustering of queries by index. □

.8 Sidelnikov-Shestakov Attack

In [SS92], Sidelnikov and Shestakov proposed an efficient attack (SiS) for recovering an alternative secret key in the McEliece cryptosystem that uses Generalized Reed-Solomon (GRS) codes. This attack exploits the structured nature of the systematic encoding generator matrix of GRS codes, which is characterized by the form $[I|A]$ where I is the identity matrix and A is a Cauchy matrix. The structure of this matrix enables the formulation of a rational function of degree one in terms of the private parameters, facilitating the recovery of these parameters through the resolution of linear equations.

Unlike in typical GRS codes, our construction samples curves that form degree-$d\kappa$ Reed-Solomon codes. The evaluation points for these polynomials are selected as a random subset of $\mathbb{F}$, and importantly, the sequence of these points is randomized before being sent to the server. This modification fundamentally alters the encoding process, deviating from the predictable pattern exploited by the SiS attack.

Furthermore, an additional layer of security is provided by encrypting the values stored on the server, which means that these values do not directly represent polynomial evaluations. This encryption obscures any structural properties that could otherwise be leveraged in an attack.

The randomized ordering and the selection of evaluation points, coupled with the encryption of codeword values, breaks the continuity and predictability required for the SiS attack to succeed. Each transmitted codeword effectively appears as if derived from a different generator matrix, due to the randomized order and content. Therefore, we conclude that our scheme is resilient against SiS attack, as well as other efficient attacks for GRS code, e.g. Schur square attack and Wieschebrink's squaring attack.