Doubly-Efficient Secret-Key PIR with Low Storage Overhead

Caicai Chen* Yuval Ishai† Aayush Jain‡

Tamer Mour§ Alon Rosen¶ Chaoping Xing‖

Abstract

In secret-key private information retrieval, a client with a short secret key retrieves a database item while hiding the requested index, and possibly also the database, from the server. The server answers using an encoded version of the database, generated via one-time preprocessing. Secret-key PIR provides an attractive "stateless" alternative to stateful PIR and oblivious RAM, and can be viewed as strengthening the standard notion of searchable symmetric encryption by not allowing any access pattern leakage.

We give the first candidate *doubly-efficient* secret-key PIR schemes that achieve a *constant* multiplicative storage overhead, asymptotically approaching 1 in natural regimes, together with $k^{o(1)}$ communication and online server work for a database of size k. The best previous online server work with constant storage overhead was $k/\mathrm{polylog}(k)$.

Our constructions follow the permuted-code blueprint for doubly efficient sk-PIR (Boyle-Ishai-Pass-Wootters and Canetti-Holmgren-Richelson, TCC 2017), and are based on similar assumptions. The main novelty is that we instantiate this blueprint using new families of "t-smooth" locally decodable codes with improved tradeoffs between rate, locality, and smoothness. This includes a new t-smooth local decoder for Reed-Muller codes using concatenated curves, as well as a construction based on curve-lifted codes that has attractive concrete efficiency features.

We perform extensive cryptanalysis of the underlying assumptions and benchmark performance under realistic parameters, demonstrating the practicality of our schemes. A representative instantiation encodes a 37 GB database of 18-bit records with only 4.2× storage overhead, while requiring the server to read less than 600 KB from the encoded database per query.

*Bocconi University and Bending Spoons. E-mail:caicai.chen@unibocconi.it.

†Technion and AWS. This work is not associated with Amazon. Supported by Israel-China ISF-NSFC grant 3127/23, ISF grant 3527/24, and BSF grant 2022370. E-mail: yuval.ishai@gmail.com.

‡Carnegie Mellon University. Supported by an NSF CAREER, an Alfred P. Sloan Fellowship, and an Amazon Research Award. E-mail: aayushja@andrew.cmu.edu.

§The Italian Institute of Artificial Intelligence. E-mail: tamer.mour@ai4i.it.

¶Bocconi University, BIDSA. Work supported by European Research Council (ERC) under the EU's Horizon 2020 research and innovation programme (Grant agreement No. 101019547) and Cariplo CRYPTONOMEX grant. E-mail: alon.rosen@unibocconi.it.

‖Shanghai Jiaotong University. Supported by Israel-China ISF-NSFC grant with the NSFC grant number 12361141818. E-mail: xingcp@sjtu.edu.cn.

Contents

.

1 Introduction

Private information retrieval (PIR) allows a client to read the i-th bit from a database $x \in \{0,1\}^k$ while hiding i from the server storing x. Multi-server PIR was introduced by Chor, Goldreich, Kushilevitz and Sudan [CGKS95] and single-server PIR by Kushilevitz and Ostrovsky [KO97].

In the original PIR model, the server must read every bit of the database, since if x_i is not read then the server knows that x_i was not queried. Motivated by the goal of PIR with sublinear server computation, Beimel, Ishai and Malkin [BIM00] proposed PIR *with preprocessing*, where the server stores an encoding $\widehat{x}$ of x computed once in an offline phase. In the online phase, a stateless client can make an arbitrary number of PIR queries, where for each query the server probes $o(k)$ bits from $\widehat{x}$ and communicates $o(k)$ bits. Such protocols are referred to as *doubly efficient PIR*.

A recent breakthrough of Lin, Mook and Wichs [LMW23] realized doubly efficient PIR under Ring-LWE. However, this protocol remains impractical due, in part, to a large storage overhead.

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

In recent work, Chen, Ishai, Mour, and Rosen [CIMR26] propose sk-PIR protocols that minimize the server's *circuit size* under new variants of the learning subspace with noise (LSN) assumption [DKL09]. Their results exhibit different tradeoffs, but achieving even slightly sublinear server probe complexity, such as $k/\mathrm{polylog}(k)$, requires superlinear storage $k^{1+\varepsilon}$.

Our starting point is the observation that in spite of significant research on doubly efficient PIR in different models, none of the candidates proposed in the literature offers practically appealing tradeoffs between server storage and online server work.[1] This applies to the "stateless" model from [BIM00], where neither the client nor the server needs to update its states; see Section 1.3 for discussion of alternative models and their disadvantages.

The initial candidates from [BIPW17, CHR17], based on secretly permuted Reed-Muller (RM) codes, are simple to describe and implement, but they suffer from a large storage overhead due to the small rate of RM: for any number of variables $m \geq 2$, using m-variate polynomials for the RM encoding reduces the communication and online work to $O(k^{1/m})$ but increases the storage by a factor of $\Omega(m! \cdot \lambda^m)$, for a security parameter λ. Even for $m = 2$, the storage overhead is quite high both asymptotically and concretely.

This motivates the search for sk-PIR schemes with better tradeoffs between communication, server online computation, and storage overhead. Are there doubly efficient sk-PIR schemes with a positive database encoding rate, namely with a constant multiplicative storage overhead?

1.1 Our Results

We present the first candidates for secret-key doubly-efficient PIR (sk-DEPIR) schemes that simultaneously achieve positive database encoding rate and k^{ε} communication and online server work (i.e., probe complexity) for a database of size k and arbitrarily small constant $\varepsilon > 0$. In fact, in one of our constructions the server work is $k^{o(1)}$ and the asymptotic rate approaches 1. Moreover,

[1] Here we refer mostly to the single-server setting. Promising progress on 2-server DEPIR has been recently made by Henzinger and Ragavan [HR26], though this construction has a slightly super-constant storage overhead. See Section 1.2 for a concrete comparison.

based on our security analysis, the concrete efficiency tradeoffs achieved by the new schemes are meaningful for practically relevant parameters.

At a high level, we follow the initial sk-DEPIR blueprint from [BIPW17, CHR17] of encoding the database using a secretly permuted linear code, but change the underlying code. We use the LSN viewpoint from [CIMR26] to guide the design and security intuition. In all cases, we use a family of *t-smooth* locally decodable codes (LDC), namely codes whose decoders have the property that any t queried positions are independent of the decoded index. Here one can think of t as a computational security parameter. The secret key selects a random code from the family, the database is encoded under this code and masked by a pseudorandom pad, and each query returns the few codeword positions needed for local decoding after unmasking. See Section 2 for more details and optimizations.

Instantiated with RM codes, this blueprint was analyzed in several prior works, and conjectured to resist polynomial-time attacks [BIPW17, CHR17, BHW19, BW21, BHMW21, CGG$^+$26]. However, as discussed above, the storage overhead of this approach is super-constant in theory and is too large for practical purposes.

As our core technical contribution, we give two new constructions of t-smooth LDCs, supporting positive encoding rate with super-constant t. Our constructions achieve query complexity k^ε for any $\varepsilon > 0$, and $k^{o(1)}$ in certain regimes. These results are unconditional, and may find other applications beyond secret-key PIR. For example, they significantly improve over previous solutions for information-theoretic PIR in a peer-to-peer setting, previously considered in [IKOS04].

The first construction is inspired by a general LDC template of Cramer, Xing, and Yuan [CXY20], but modifies it to use a concatenated code for decoding. It can achieve positive rate and scales well with the smoothness parameter t, but it cannot achieve rate approaching 1. The security of the sk-DEPIR resulting from this construction reduces to a concatenated-codes adaptation of the assumptions underlying prior work [BIPW17, CHR17].

Our second construction is based on a new instantiation of a "lifted code" construction of Guo, Kopparty, and Sudan [GKS13]. Its smoothness parameter t needs to grow more slowly with k, but is good enough for asymptotic rate 1 with super-constant t, as well as for concrete efficiency. The local decoder in this construction is the same as the RM decoder, but the specific choice of parameters requires the sk-DEPIR to add dummy queries for security. The security of this scheme reduces to a close variant of a previous conjecture from [CHR17], and is in fact implied by an extension of the conjecture to the parameter regime relevant to our construction. Thus, we obtain the stronger asymptotic conclusion under a similar assumption.

Our constructions support both an asymptotic theory-oriented regime in which we choose t to be super-constant in k, typically polylogarithmic, and an applied "fine-grained" regime aimed at large k in which we take t to be a small constant, typically $t \geq 4$, to enable good concrete efficiency.

As a last contribution of this work, we complement our constructions with cryptanalysis of the underlying hardness assumptions. We substantially expand asymptotic cryptanalysis from prior work [BIPW17, CHR17, BHW19, BW21, BHMW21, CGG$^+$26] and extend it to the assumptions new to this work. In addition, we provide concrete cryptanalysis, enabling us to derive concrete security evaluation of these assumptions, and estimates to guide our concrete parameter choices.

We examine the runtime of natural algebraic and statistical attacks, and identify the most effective ones in different parameter regimes. We prove unconditional asymptotic lower bounds for the complexity of some of these attacks, and give conjectured concrete lower bounds based on heuristics, supported by empirical evidence and/or well-established literature.

1.2 Concrete Efficiency

We demonstrate that our constructions can be instantiated to yield the first doubly-efficient sk-PIR schemes that offer improvements for practically relevant parameters. All benchmarked instances are chosen to target at least 100 bits of conjectured security, which we derive by extensive concrete cryptanalysis (Section 6).

Our most practically attractive scheme is the one based on lifted codes. It can be instantiated on a 20 GB size database of 18-bit records, while using less than 155 GB storage to encode it (7.7× overhead), and requiring the server to probe $\approx 2^{18}$ records in the encoded database, which is ≈ 34,000× less than scanning the (original) database. The client's online runtime is benchmarked at less than 60 ms per query, when implemented in Go and run on a Macbook M5 Pro.

Encoding the database under the lifted code is performed in two phases: First, a one-time pre-processing phase for computing the parameters of the code, that can be re-used across different databases. Second, a database-dependent encoding, which falls in the same broad order of magnitude as a Reed–Solomon encoding at the same scale. While we do not benchmark database encoding, we roughly estimate each of the phases to take 0.5 – 3 days for the above 20 GB database on a single modern high-end GPU with large, fast storage, by interpolating existing benchmarks of similar computations [Man14, HMMM14, NVI10, NVI26]. Notably, this computation is highly parallelizable, offering near-linear efficiency gains with additional computation units, provided sufficient memory and storage bandwidth.

For comparison, based on our analysis, the original sk-DEPIR schemes from [BIPW17, CHR17] based on RM codes could only support a database that is ≈ 13× smaller (1.57 GB) with the same storage and probe complexity, for the same level of security.

It is also instructive to compare the above with the alternative[2] 2-server model. The recent information-theoretic DEPIR scheme from [HR26], on an 11 GB database with 8-bit records, has storage of 1 TB per server (180× overhead in total). Each server needs to probe $\approx 4.4 \cdot 2^{20}$ records, which is ≈ 2,560× less than scanning the database.

Even better storage overhead of 6× is obtained by our other sk-PIR construction, based on concatenated decoding, for larger database sizes of 150 GB and more, though this comes at a higher cost in probe complexity and communication, and security based on assumptions new to this work.

We also consider a setting where the adversarial server is bounded in the number of queries it observes. In this setting, we conjecture a similar security level to hold with smaller smoothness parameters, which yields further efficiency gains. For a realistic bound of 2^{50} observed queries, the lifted-code construction supports a 36.6 GB database over 18-bit records using the same 155 GB storage from above (4.2× overhead), while still requiring the server to read only $\approx 2^{18}$ records per query, about 62,000× less than scanning the original database. A ballpark estimate for encoding the 36.6 GB database is 1 – 5 days, for each of the one-time parameters computation and the per-database encoding, on a single modern GPU with high-bandwidth storage. At a larger scale, it supports a 712 GB database over 20-bit records using 2.75 TB storage (3.9× overhead), while requiring the server to read only $\approx 2^{20}$ records per query, about 270,000× less than scanning the original database.

Full benchmarks and further implementation details are given in Tables 1 and 2 and Section 7.

[2]While 2-server PIR is technically incomparable to sk-PIR, the latter implies a variant of the former. Indeed, if one server stores the secret key and the other stores the encoded database, the role of the sk-PIR client can be distributed between the client and the key server. Unlike the standard 2-server model, here only one server needs to store the encoded database, and the key server can be distributed across any number of servers to further improve the trust model without affecting the asymptotic complexity.

The source code is available at `https://github.com/Caicai-Chen/FastSecretStatePIR`.

While we do not obtain practically feasible results in the public-key setting, we argue that sk-DEPIR with a fast distributed client captures many of the same deployments of interest. In fact, in uses cases of PIR in which the database is encrypted and is not owned by a single client, even the public variant of DEPIR requires a distributed decryption process. The same trust model can be used for distributed encryption of queries in sk-DEPIR. See [CIMR26, BCH+25] and Section 1.3.

We note that one can, in principle, obtain a public-key variant by obfuscating the secret-key client algorithm, that maps an index i to the corresponding query and decoding information, so that anyone can generate valid queries without learning the secret key [BIPW17]. Thus, in principle, our new asymptotic efficiency tradeoffs can apply also in a public-key setting. However, this approach is not yet practical (except via the use of trusted hardware) and is secondary to our main focus on secret-key constructions and their efficiency.

1.3 Alternative Models

Secret-key preprocessing naturally supports settings where the database must be kept secret from the server, since the database encoding can also encrypt or mask the database. This lets doubly-efficient sk-PIR act as a de-facto mechanism for *private search over encrypted data*, aiming to avoid query leakage beyond what is inherent in correctness.

This contrasts with the traditional notion of *searchable symmetric encryption* (SSE), where efficient schemes typically come with explicit leakage profiles across multiple queries, including access-pattern leakage and related correlations [SWP00, CGKO06]. In applications where preventing such leakage is a primary goal, such as searching data structures, PIR-style query privacy provides a different design point, and sk-PIR was highlighted as a natural candidate for achieving it in [CIMR26].

Stateful single-server PIR systems that achieve sublinear server work typically rely on a query-dependent client and/or server state that is initialized in a preprocessing stage and then stored and updated across executions, often to enable reshuffling, amortization, or the consistency of auxiliary data structures [PPY18, CK20, KC21, ZPSZ23, MIR23, LP24]. This statefulness can complicate concurrency, recovery, and multi-device use, and it can impose additional requirements on deployment and correctness when queries are interleaved or interrupted.

Oblivious RAM [GO96] similarly hides access patterns by continuously re-randomizing the server-side layout, but it inherently requires maintaining and updating a structured state and typically incurs logarithmic overhead in bandwidth and server work per logical access, as well as super-constant round complexity. In contrast, sk-PIR is stateless on both sides beyond the server-stored encoding and the client-held key, it only involves a single round of queries, and it natively supports many concurrent queries to the same stored encoding without per-query state updates.

To summarize, the advantage of sk-PIR over stateful PIR and ORAM is that it achieves PIR-style query privacy without requiring ongoing state maintenance or reshuffling across queries, which simplifies concurrent and distributed deployments.

There has also been a significant body of work on optimizing the asymptotic and concrete efficiency of DEPIR with deterministic (keyless) database encoding. In the single-server setting, following the theoretical breakthrough of [LMW23], subsequent optimization efforts could not yield truly practical solutions [OPPW23, XZO+26]. Lower bounds on the extra storage were given in [BIM00, PY22]. In the multi-server setting, following the initial constructions from [BIM00], improved schemes were given in [GLM+24, LLFP24, HR26]. A hybrid 2-server PIR model in which only one of the servers has sublinear computation was recently considered in [AS26]. See Section 1.2 for a comparison of our sk-DEPIR schemes with the 2-server DEPIR scheme from [HR26].

Finally, although our schemes are cast the secret-key setting, the client can be distributed among two or more parties, which jointly generate queries and decode answers using a lightweight MPC protocol.[3] This eliminates a single point of failure and can match non-collusion assumptions that are already present in applications. For example, non-collusion assumptions are necessary when PIR is invoked by an MPC protocol, or when the database itself is secret and is not owned by a single client. In such settings, sk-DEPIR with a fast distributed client can provide the same functionality as public DEPIR with the same trust model. The only disadvantage is that one needs to distribute not only the decryption of the answers but also the encryption of the queries.

2 Technical Overview

In this section we describe the main ideas behind our improved constructions, starting with the high-level approach.

2.1 Blueprint: sk-DEPIR from Smooth LDCs

Our starting point is an abstraction of the initial doubly-efficient sk-PIR schemes from [BIPW17, CHR17]. These constructions are implicitly based on a *smooth* locally decodable code (LDC), allowing a recovery of the i^{th} element in an encoded database $\mathbf{x}$ by querying few locations in its encoding $\mathbf{c} = \mathsf{E}(\mathbf{x})$. Crucially, for a parameter t that grows with the security parameter, the query sequence is t-smooth, in the sense that any t indices in this sequence have marginal uniform distribution, and in particular, are jointly independent of i.

A natural approach to sk-DEPIR could be to simply let the server store an encoding of the database under such a smooth LDC. In order to fetch an entry $\mathbf{x}_i$, a client could send to the server a random set of locations $Q = (y_1, \ldots, y_\ell)$ that allows a t-smooth recovery of $\mathbf{x}_i$. The hope is that the information-theoretic notion of t-smoothness in the query set translates into computational hardness that implies security. Indeed, t-smoothness is already sufficient, and as observed in [BIPW17] it is also necessary, to rule out a class of *local attacks*.

Unfortunately, the above simple construction is typically insecure. Whenever the LDC is linear (which is the case for almost all existing LDCs) it will be subject to a simple *linear-algebraic attack*: the server can simply check if the set of linear combinations of the database entries defined by the query set *spans* a given target entry of the database, breaking the scheme in polynomial time.

The above attack crucially relies on the knowledge of the linear code. The idea of [BIPW17, CHR17] was to circumvent the attack by hiding the code from the server, namely by using a *random* code sampled from a *family* of LDCs. This gives rise to a solution in the secret-key setting, where the client derives the code using a secret key.

A simple way to obtain a large family of LDCs is to fix a t-smooth LDC and consider the $n!$ codes induced by permuting the codeword coordinates (here, n is the length of the code). Indeed, [BIPW17, CHR17] instantiate this template using *Reed-Muller codes* (Definition 3.5), which support t-smooth decoding via random parametric curves [BF90, GLR$^+$91, GS92].

More concretely, the server stores an encrypted systematic RM encoding of the database, where the coordinates are permuted by a hidden pseudorandom permutation π generated using the client's secret key. Recall that a systematic RM encoding of $\mathbf{x}$ over a finite field $\mathbb{F}$ is the evaluation table of a low-degree m-variate polynomial $\mathbf{c} : \mathbb{F}^m \to \mathbb{F}$ such that $\mathbf{c}(i) = \mathbf{x}_i$ (for an a-priori fixed embedding of

[3] A simpler alternative is to run the client inside a trusted execution environment (TEE). Opting for an MPC-based distributed client, concrete efficiency can be greatly improved by employing MPC-friendly PRFs (e.g., [BIP$^+$18]) and simpler families of permutations, e.g., alternating between cyclic shifts and bit-level permutations a constant number of times. We leave the optimization and analysis of such distributed client protocols to future work.

$[k]$ in $\mathbb{F}^m$). To retrieve the i^{th} database entry, the client samples a uniformly random degree-t curve that passes through point i, namely a tuple $\psi = (\psi_1, \ldots, \psi_m)$, where each $\psi_j(z)$ is a univariate polynomial of degree (at most) t and $\psi(0) = i$. The client sends a sufficiently large random subset of the points $\{\psi(z) \mid z \neq 0\}$, renamed using π, to the server, requesting the codeword values at these locations. Under an appropriate choice of parameters, the client is able to recover $\mathbf{x}_i$ from these values via univariate polynomial interpolation.

The security of the constructions is then based on the assumption that random sub-samples from random low-degree curves are indistinguishable from uniformly random query sets, when the elements of $\mathbb{F}^m$ are re-labeled under a hidden random permutation. Boyle et al. [BIPW17] call the corresponding problem the *permuted low-degree puzzle* (PLD, Definition 5.3) and conjectures that it is hard. Canetti et al. [CHR17] propose an additional security measure where the query set is mixed with a number of uniformly random points, obtaining their version of the assumption which they referred to as *hidden permutation with noise* (HPN). We consider an arguably more natural variant which we call the *permuted low-degree with noise* problem (PLDN, Definition 5.4), presented in Fig. 1. PLDN implies sk-DEPIR by a direct adaptation of the constructions from [BIPW17, CHR17] and is at least as hard as PLD and HPN under a suitable choice of parameters.[4]

(m, t, ℓ, L)-PLDN_q

Parameters:

- Finite field $\mathbb{F}_q$.
- Degree t, number of variables $m \geq 2$, number of points ℓ and sample size $L \geq \ell$.
- Uniformly random permutation $\pi : \mathbb{F}_q^m \to \mathbb{F}_q^m$.

$\mathsf{PLDN}_\pi(i \in \mathbb{F}_q^m)$:

1. Sample a uniformly random degree-t parametric curve $\psi : \mathbb{F}_q \to \mathbb{F}_q^m$ such that $\psi(0) = i$.
2. Sample uniform and distinct $z_1, \ldots, z_\ell \leftarrow \mathbb{F}_q^*$.
3. Let $y_j = \pi(\psi(z_j))$ for $j = 1, \ldots, \ell$ and sample $y_{\ell+1}, \ldots, y_L \leftarrow \mathbb{F}_q^m$ uniformly at random.
4. Sample a uniformly random permutation $f : [L] \to [L]$.
5. Output $y_{f(1)}, \ldots, y_{f(L)}$.

Figure 1: The permuted low-degree with noise (PLDN) distribution (Definition 5.4). The PLDN conjecture (Conjecture 5.2) asserts that $\mathsf{PLDN}_\pi(i)$ samples corresponding to polynomially many adversarially chosen indices i are indistinguishable from sequences of uniformly random locations (with replacement) of appropriate length, whenever $L > \ell + \lambda$, $t = \omega(1)$ and $q = \lambda^{\Omega(1)}$. PLDN implies the security of RM-based sk-DEPIR [BIPW17, CHR17] and the security of our construction based on lifted RS codes (Theorem 2.4). PLDN is implied by the conjectures from [BIPW17, CHR17] under relevant choices of parameters.

While the RM-based instantiation may superficially seem promising in terms of concrete efficiency, it suffers from a high storage overhead. Concretely, the rate of the code is roughly $\frac{1}{m! \cdot t^m}$ where m is the number of variables. In this setting, the total communication is $O(m \cdot k^{1/m})$ where k is the size of the database. This presents us with the following tension: for security one requires that t grows with the security parameter, and for sublinear communication we need $m \geq 2$. This inherently rules out a constant storage overhead that does not grow with the security level, and makes the $m! \cdot t^m$ storage overhead prohibitive in practice.

[4]The difference between HPN and PLDN is that in HPN the curve is evaluated over a fixed set of field elements and the noise coordinates are always at fixed locations in all samples, while in PLDN both the noise locations and the evaluation points are sampled uniformly at random for each sample. See further discussion at the end of Section 5.2.

To address this tension, we propose two new instantiations of the above framework by introducing two new families of t-smooth LDCs. These codes reduce the dependency between the rate of the code and the smoothness parameter t. While the first code we present achieves rate $1/m!$ (independent of t) at the cost of making query complexity $O(t \cdot m \cdot k^{2/m})$, the second one achieves a rate approaching 1, with query complexity $O(m \cdot k^{1/m})$ for moderate super-constants t, m. This provides the first candidate sk-DEPIR schemes simultaneously achieving query complexity $k^{o(1)}$ and constant server storage overhead.

Other techniques from the literature on (1-smooth) LDCs, such as LDCs from multiplicity codes [KSY11, Yan25] and expander-based LDCs [HOW15] do not directly generalize to yield attractive parameters in the t-smooth setting, though it is possible that combining these techniques with ours can lead further improvements. We leave a more systematic exploration of this space to future work.

2.2 First LDC Construction: RM Decoding Along Concatenated Curves

Low-degree curves are useful for decoding Reed-Muller codewords due to their special algebraic structure: one can think of a low-degree m-variate curve as a tuple of Reed-Solomon codewords (Definition 3.3), one over each of the m different variables. RS codewords have the property that they are *multiplication-friendly*: The pointwise product of RS codewords is itself a redundant RS codeword (of a higher degree). This property by itself is already sufficient to locally decode a RM codeword: Since a RM codeword, by definition, is a linear combination of low-degree monomials over its m variables, the values of such a codeword over low-degree curves correspond then to a "product" RS codeword, which is redundant and therefore decoding is possible.

Letting $*$ denote pointwise product, the local decoding argument for Reed-Muller using multiplication-friendly codes, in this case – Reed-Solomon, may be summarized as follows:

$$\begin{aligned}
&\mathbf{c} \in \mathsf{RM} \quad \Longrightarrow \quad \mathbf{c}(z_1, \ldots, z_m) = \sum_{j_1,\ldots,j_d} \gamma_{j_1,\ldots,j_d} \cdot \prod_i z_{j_i}, \\
&\psi_j \in \mathsf{RS} \quad \Longrightarrow \quad \mathbf{c} \circ \psi \in \mathbf{Span}\big(\{\psi_{j_1} * \cdots * \psi_{j_d}\}\big) \in \mathsf{RS} * \cdots * \mathsf{RS} \subseteq \mathsf{RS}',
\end{aligned}$$

where $\psi = (\psi_1, \ldots, \psi_m)$ is a degree-t curve, RS is the corresponding degree-t Reed-Solomon code, and RS' is a Reed-Solomon code with a higher dimension $\dim(\mathsf{RS}') \geq d \cdot \dim(\mathsf{RS})$.

We construct a new local decoder for RM with improved trade-off between smoothness and rate by relying on *concatenated RS codes*, which are also multiplication-friendly.

Concatenated RS codes are obtained by composing two RS codes: an inner RS code over a base field $\mathbb{F}_q$ and an outer RS code over an extension field $\mathbb{F}_{q^e}$ ($e = 2$ will suffice for our purposes). Roughly speaking, a codeword consists of codewords from the inner code, each encoding a symbol from an underlying outer-RS codeword. (To encode a message, first encode it using the outer code, then embed each $\mathbb{F}_{q^e}$-symbol as a vector in $\mathbb{F}_q^e$ and encode it using the inner code.)

The local decoder using concatenated RS codes (Fig. 4) samples points on a "concatenated curve," namely a query set defined by an analogous inner/outer curve structure, and its correctness follows from the multiplication-friendliness of concatenated RS by the above reasoning.

The concatenated-curve decoder allows us to break the dependency between the smoothness parameter t and the rate of the code from which RM codes suffer: the constraint over t that ensures correct decoding now applies to the much larger extension field. On the flip side, the concatenated structure of the query set is not at all smooth, but rather satisfies certain relaxations of the notion, which are still sufficient for plausible security of the obtained sk-PIR construction (as we discuss in Section 2.4).

Theorem 2.1 (RM with Concatenated-Curve Decoding, Informal (Thm. 4.1, Cor. 4.2)). *For any $\varepsilon > 0$, there exists a weakly t-smooth LDC with $t = \Omega(k^{\varepsilon})$, positive rate, and locality $\ell = O(k^{3\varepsilon})$.*

2.3 Second LDC Construction: Curve-Lifted Reed-Solomon Codes

In our second new construction of smooth LDCs, we stick to the same curve-based local decoder of RM codes but modify the way in which the database is encoded. It turns out that RM codewords, namely low-degree multivariate polynomials, are merely a strict subset of all encodings that can be locally decoded using low-degree curves; we can in fact encode more information with the same decoder, achieving a much higher rate in certain parameter regimes.

We refer to *curve-lifted Reed-Solomon codes* (lifted RS for short), which are defined to consist of any codeword $\mathbf{c} : \mathbb{F}_q^m \to \mathbb{F}_q$ for which every restriction to a low-degree curve is a Reed-Solomon codeword. Lifted RS codes are locally decodable immediately by design: Since any restriction to a curve is a RS codeword, it is in particular redundant, therefore reading a sufficiently large subset of its coordinates allows to recover it all. Hence, the RM local decoder that reads from a punctured low-degree curve passing through the target coordinate works for lifted RS as well.

Two important questions then remain: First, what rate can we achieve with these codes? Second, can we efficiently compute a systematic lifted-RS encoding of a given database?

The Rate of Lifted RS. The work of [GKS13] showed that, for $t = 1$, lifted RS codes achieve rate approaching 1 for any $m \geq 2$. Subsequent works further analyze the rate of RS codes lifted to lines ($t = 1$) [HPPV20], to quadratic curves ($t = 2$) [LHP+21], or to other specific algebraic structures (so-called "η-lines") [LN21]. We extend this analysis to work with general values of t and m that can even be small super-constants.

Theorem 2.2 (Lifted-RS LDC, Informal (Thm. 4.2, Cor. 4.3)). *For any $\varepsilon > 0$, there exists a t-smooth LDC with $t = \Omega(\log \log k)$, rate approaching 1, and locality $\ell = O(k^{\varepsilon})$.*

Alternatively, for any $t = o(\log \log k)$, such codes exist with rate approaching 1 and locality $\ell = k^{o(1)}$.

Lifted-RS is *affine-invariant* by definition, namely it is invariant under affine transformations over its coordinate space. Indeed, an affine transformation of a low-degree curve is still a low-degree curve. A known fact (see, e.g., [HRS13]) is that any affine-invariant code can be written as the span of a basis of monomials. Namely, there exists a set of "good monomials" $\mathcal{G} = \{M_{e_1,\ldots,e_m}(\alpha_1, \ldots, \alpha_m) := \prod_i \alpha_i^{e_i}\}$ such that any lifted RS codeword $\mathbf{c}$ may be written as

$$\mathbf{c}(z_1, \ldots, z_m) = \sum_{M \in \mathcal{G}} \gamma_i M_i(z_1, \ldots, z_m).$$

While RM (which is also affine-invariant) is spanned by all low-degree monomials, the basis $\mathcal{G}$ for lifted RS contains monomials of arbitrary degrees, as long as their restriction to low-degree curves is an evaluation of a low-degree polynomial (i.e., a RS codeword). The rate of a lifted RS code, then, is precisely $|\mathcal{G}|/q^m$ and, following prior work, it is sufficient to lower-bound $|\mathcal{G}|$.

We characterize the monomials in $\mathcal{G}$ by identifying a "goodness" criterion over exponent vectors $(e_1, \ldots, e_m) \in \{0, \ldots, q-1\}^m$ that leads to low-degree curve-restrictions. One limitation in our analysis is that it only considers the case where these restrictions are of degree at most $q-2$, corresponding to a lifting of the RS code of dimension $q-1$; this is the maximum dimension that still allows local decoding, and where to recover a codeword coordinate one must read *all* other coordinates on a curve that contains it.

We show that the goodness criterion can be tested using a $\tilde{O}(q \cdot tm)$-time dynamic programming algorithm (Algorithm 1). This immediately gives us a way to empirically estimate the rate of a lifted RS code: Sample monomials uniformly at random and test whether they belong to $\mathcal{G}$. The fraction of good monomials in the samples provides a concrete Monte-Carlo estimation of the rate. We present rate estimates for different parameter choices in Table 3, showing that lifted RS yields better rate over RM by a factor of at least 10 for the parameter ranges most relevant to our benchmarks (Table 1).

We additionally prove an explicit, though quite crude, lower bound on the asymptotic rate of our lifted RS codes (with RS dimension $q-1$), implying that the rate approaches 1 with $t = \Omega(\log \log k)$ (Theorem 2.2). We invoke a "forbidden-pattern" counting argument to upper bound the fraction of *bad* monomials, in the spirit of other rate analyses for lifted codes, e.g., [GKS13, HPPV20, LN21].

Systematic Encoding of Lifted RS. Being a linear code, there is a generic polynomial-time encoder for lifted RS using a generator matrix. While computing the generator matrix can be done in time $\tilde{O}(q^{m+1} \cdot tm) = \tilde{O}(k^{1+1/m})$ by running our goodness test algorithm (Algorithm 1) over all possible monomials, generic systematic encoding given the matrix takes time $\Omega(k^3)$, and quickly becomes impractical as the database size grows.

Instead, we use the affine-invariant structure underlying lifted RS codes to obtain a practical quasi-linear time systematic encoding algorithm. The concrete complexity of the encoder is comparable to that of encoding a RS code with the same dimension and length, up to a small multiplicative constant, assuming a one-time database-independent preprocessing step.

A key observation from [GK16] is that, when identifying the coordinate space $\mathbb{F}_q^m$ with the extension field $\mathbb{F}_{q^m}$ by an $\mathbb{F}_q$-linear isomorphism $\varphi : \mathbb{F}_{q^m} \to \mathbb{F}_q^m$, we obtain a code $C' \subset \{\mathbb{F}_{q^m} \to \mathbb{F}_q\}$ defined by:

$$C' = \{\mathbf{c} \circ \varphi \mid \mathbf{c} \in C\}.$$

It turns out that C' is also affine-invariant by the linearity of φ, and is therefore spanned by a basis of *univariate* monomials: it can be characterized by a set of degrees $D' \subset \{0, \ldots, q^m - 1\}$ such that $C' = \mathbf{Span}(\{\mathbf{c}(z) = z^i \mid i \in D'\})$. We can efficiently find these monomials by looking at the support of sufficiently many random codewords. Another useful observation from [GK16] is that the set $S = \{\omega, \omega^2, \ldots, \omega^k\}$, for any generator $\omega \in \mathbb{F}_{q^m}^*$, constitutes an interpolation set for C' (i.e. $\dim(C'|_S) = k$).

Viewing the code C through the lens of the univariate code C' suggests the following algorithm for systematically encoding a database $\mathbf{x} \in \mathbb{F}_q^k$:

1. Find $\gamma_1, \ldots, \gamma_k \in \mathbb{F}_{q^m}$ satisfying $\sum_{i \in D'} \gamma_i (\omega^j)^i = \mathbf{x}_j$ for all $j \in [k]$.
   Such a solution exists since S is an interpolation set and it can be found by solving a generalized Vandermonde linear system.

2. Evaluate the polynomial $\mathbf{c}'(z) = \sum_{i \in D'} \gamma_i z^i$ over all $z \in \mathbb{F}_{q^m}$.

3. Embed $\mathbf{c}'$ back to the original coordinate domain $\mathbb{F}_q^m$. Namely output the codeword $\mathbf{c} : \mathbb{F}_q^m \to \mathbb{F}_q$ defined by $\mathbf{c} = \mathbf{c}' \circ \varphi^{-1}$.

The first two steps can be performed in quasi-linear time using standard FFT techniques. The last step also takes quasi-linear time using an isomorphism that can be computed in time $\tilde{O}(1)$ when m is constant.

2.4 The Resulting sk-DEPIR Schemes and Their Security

We now plug in our two high-rate smooth LDC constructions in the blueprint from [BIPW17, CHR17] to obtain sk-DEPIR with low storage overhead.

In both instantiations, the client encodes the database using the respective t-smooth LDC, for a smoothness parameter t that affects the conjectured security of the scheme. The encoding is then permuted by the client using a pseudorandom permutation π. At query time, the client uses the corresponding local decoder to retrieve any database element. Specifically, the client computes a set of ℓ decoding locations using the local decoder, permutes the locations under π, then possibly mixes them with uniformly random coordinates in a set of total size $L \geq \ell$.

Secret-Key DEPIR from Concatenated-Curve Decoding. Reed-Muller codes, equipped with the concatenated-curve local decoder (Theorem 2.1), allow us to encode the database in constant rate for any polynomially-bounded (weak) smoothness parameter $t := t(\lambda)$.

Every query in the server's view in the resulting sk-DEPIR consists of a subset of coordinates in a concatenated curve, permuted under a hidden random permutation and mixed with noise. We explicitly describe this distribution, dubbed *permuted concatenated curves with noise*, in Fig. 2. The security of the sk-DEPIR is based on the conjecture that such view hides the client's requested indices. This conjecture can be viewed as an adaptation of the PLDN conjecture where the uniformly random low-degree curves are replaced by a uniformly random *concatenated* low-degree curves. It stems from the heuristic that applying an "inner RS encoding" over the outer-curve symbols before the random permutation does not induce computationally detectable structure.

In particular, even though concatenated-curve decoding satisfies only a weak notion of smoothness, where any t coordinates are statistically independent of the requested index but are not necessarily uniform (Definition 4.1), local attacks still fail since the distribution satisfies "smoothness on average": random restrictions of the distribution, say to $\sqrt{t}$ coordinates, are smooth with high probability (see Section 6.2.6 for further details).

Theorem 2.3 (sk-DEPIR from Permuted Concatenated Curves, Informal (Thm. 5.3))**.** *For any $\varepsilon > 0$, assuming the permuted concatenated curves with noise conjecture, there exists sk-DEPIR with constant storage overhead and $O(k^{\varepsilon})$ communication and server work.*

We conjecture that the obtained scheme is secure against polynomial-time adversaries whenever the underlying RM code is instantiated with $t = \omega(1)$ and $q = \lambda^{\Omega(1)}$. In fact, we are not aware of any attack that breaks the scheme in time $q^{o(t)}$. Therefore, when $t = \mathrm{poly}(\lambda)$ (as allowed in Theorem 2.1 for large enough databases), we plausibly obtain *sub-exponential security.*

Except for extreme parameter choices (specifically when r and s from Fig. 2 are close to q and, resp., q^e by a constant; see Section 6.1.2), we are not aware of any effective attacks even when the curve coordinates are not mixed with noise, i.e. when $L = \ell$. This is akin to PLDN, where noise was introduced by [CHR17] only as an additional security measure.

Secret-Key DEPIR from Lifted RS Codes. We use lifted RS codes to instantiate the sk-DEPIR template of [BIPW17, CHR17]. Since the local decoder for lifted RS is identical to that of RM with matching parameters, the server's view in the sk-DEPIR remains identical to that of the original RM-based instantiations. Therefore, we are able to base the security of the obtained sk-DEPIR on the hardness of the PLDN problem (Fig. 1), which we conjecture holds against polynomial-time adversaries whenever $t = \omega(1)$ and $q = \lambda^{\Omega(1)}$; the best attacks we are aware of run in time $q^{o(t)}$ for field size q and smoothness parameter t and, similarly to the scheme from

(m, t, e, s, r, L)-PLDN-Conc$_q$

Parameters:

- Finite field $\mathbb{F}_q$ and its extension $\mathbb{F}_{q^e}$. We map every element $v \in \mathbb{F}_{q^e}$ to a deg-$<e$ polynomial $\widehat{v} : \mathbb{F}_q \to \mathbb{F}_q$ by the isomorphism $\mathbb{F}_{q^e} \cong \mathbb{F}_q^{<e}[\alpha]$.
- Degree t, number of variables m, number of points $\ell = s \cdot r$ and sample size $L > \ell$.
- Uniformly random permutation $\pi : \mathbb{F}_q^m \to \mathbb{F}_q^m$.

PLDN-Conc$_\pi(i \in \mathbb{F}_q^m)$:

1. Sample a uniformly random degree-t parametric curve $\psi : \mathbb{F}_{q^e} \to \mathbb{F}_{q^e}^m$ such that $\psi(0) = i$.
2. Let $\psi' : \mathbb{F}_{q^e} \times \mathbb{F}_q \to \mathbb{F}_q^m$ be defined such that $\psi'_j(z, w) = (\widehat{v}_1(w), \ldots, \widehat{v}_m(w))$ where $(v_1, \ldots, v_m) = \psi(z)$.
3. Sample uniform and distinct $z_1, \ldots, z_s \leftarrow \mathbb{F}_{q^e}^*$.
4. For $j = 1, \ldots, s$, sample uniform and distinct $w_{j,1}, \ldots, w_{j,r} \leftarrow \mathbb{F}_q^*$.
5. Let $y_{r(j-1)+h} = \pi(\psi'(z_j, w_{j,h}))$ for $j = 1, \ldots, s$ and $h = 1, \ldots, r$.
6. Sample $y_{\ell+1}, \ldots, y_L \leftarrow \mathbb{F}_q^m$ uniformly at random.
7. Sample a uniformly random permutation $f : [L] \to [L]$.
8. Output $y_{f(1)}, \ldots, y_{f(L)}$.

Figure 2: The distribution underlying the security of the sk-DEPIR construction based on concatenated-curve decoding of RM codes (Theorem 2.3). Security follows under the assumption that PLDN-Conc$_\pi(i)$ samples corresponding to adversarially chosen inputs i are indistinguishable from those corresponding to uniformly random i. Theorem 2.3 may be based on any regime where e is a small constant, $r = \Omega(q)$ and $s = \Omega(q^e)$.

Theorem 2.3, natural classes of statistical attacks provably fail unless they observe $q^{\omega(\sqrt{t})}$ samples in representative asymptotic regimes.

Theorem 2.4 (sk-DEPIR from PLDN, Informal (Thm. 5.2))**.** *For any $\varepsilon > 0$, assuming the PLDN conjecture, there exists sk-DEPIR with storage overhead approaching 1 and $O(k^\varepsilon)$ communication and server work.*

To obtain lifted RS codes with rate approaching 1 with sublinear-query local decoder, we must work with smoothness parameter $t = O(\log \log \lambda)$. We therefore get a sk-DEPIR scheme that can be broken by quasi-polynomial time adversaries (unlike the scheme from Theorem 2.3), but is still conjectured to be hard against polynomial-time adversaries.

Nevertheless, the construction from lifted RS codes offers attractive concrete trade-offs between efficiency and conjectured security, making it a promising candidate for practical sk-DEPIR. In Table 3 we provide evaluation of the storage overhead based on concrete parameter choices.

2.5 Cryptanalysis

A central contribution of this work is cryptanalysis of the hardness assumptions underlying our constructions, namely PLDN (Fig. 1) and its variant of concatenated curves (Fig. 2).

We discuss the relation between our assumptions and similar assumptions from the literature, including the "permuted puzzles" assumptions from [BHW19, BW21, BHMW21] and the more recent related assumptions from [CGG+26]. We additionally provide a unified perspective on these problems from the lens of *learning subspace with noise* (LSN) problems [DKL09, CIMR26]. This perspective provides a clean framework for examining *algebraic attacks* against our assumptions, including rank attacks, annihilating polynomials, and attacks on McEliece-style cryptosystems.

We also analyze the success of *statistical attacks* against our assumptions, including birthday attacks, SQ attacks, pairwise statistics and local distinguishers. We provide conservative estimates of the concrete complexity of such attacks, based on well-established heuristics, from which we derive the concrete parameters (Conjectures 6.1 and 6.2). Finally, we prove unconditional lower bounds on the power of these attacks that support our asymptotic conjectures and complement the concrete analysis (Conjectures 5.1 to 5.3). We refer the reader to Section 6 for more details.

3 Preliminaries

For $n \in \mathbb{N}$, we denote $[n] = \{1, \ldots, n\}$. As a convention, we use boldface notation for vectors, e.g. $\mathbf{x}$, and assume that vectors are columns by default. For a vector $\mathbf{x} \in \Sigma^n$, we use $\mathbf{x}_i$, for $i \in [n]$, to denote its i^{th} element and $\mathbf{x}_S$, for $S \subseteq [n]$, to denote its restriction to locations in S, which is a vector of length $|S|$. For a subset $A \subseteq \Sigma^n$, we use $A\,|_S$ to denote the restriction $A\,|_S = \{\mathbf{x}_S \mid \mathbf{x} \in A\}$.

We denote by $\{X \to Y\}$ the set of all functions from X to Y, i.e. Y^X, and interchangeably view vectors $\mathbf{x} \in \Sigma^n$ as functions in $\{[n] \to \Sigma\}$, consequently denoting $\mathbf{x}(i) = \mathbf{x}_i$.

We use $d_H(\mathbf{x}, \mathbf{x}')$ to denote the Hamming distance between two vectors $\mathbf{x}, \mathbf{x}' \in \Sigma^n$ and by $w_H(\mathbf{x}) = d_H(\mathbf{x}, 0^n)$ the Hamming weight of $\mathbf{x}$. We denote the point-wise product of vectors $\mathbf{x} = (x_1, \ldots, x_n) \in \mathbb{F}^n$ and $\mathbf{y} = (y_1, \ldots, y_n) \in \mathbb{F}^n$ by $\mathbf{x} * \mathbf{y} = (x_1 y_1, \ldots, x_n y_n)$.

Algebra. For a prime power q, we use $\mathbb{F}_q$ to denote the finite field of size q. We use $\mathrm{GL}_n(\mathbb{F}_q)$ to denote the general linear group of degree n over $\mathbb{F}_q$. We use $\mathbb{F}_q[\alpha_1, \ldots, \alpha_m]$ to denote the space of all m-variate polynomials over $\mathbb{F}_q$. In this work, the default notion of degree for polynomials is the *total* degree, which we denote by $\deg(\cdot)$. We let $\mathbb{F}_q[\alpha_1, \ldots, \alpha_m]_{\leq d}$ and $\mathbb{F}_q[\alpha_1, \ldots, \alpha_m]_{<d}$ denote the sets of all m-variate polynomials over $\mathbb{F}_q$ of total degree at most d and, respectively, less than d.

A *parametric curve* in m variables over $\mathbb{F}_q$ is defined by an m-tuple of polynomials $\psi = (\psi_1, \ldots, \psi_m)$, where $\psi_j \in \mathbb{F}_q[\alpha]$. We view ψ as a mapping from $\mathbb{F}_q$ to $\mathbb{F}_q^m$ where $\psi(z) = (\psi_1(z), \ldots, \psi_m(z))$. We define the degree of ψ to be $\deg \psi = \max_j \deg(\psi_j)$.

3.1 Standard Cryptographic Notions

We use λ to denote a computational security parameter and require by default security against non-uniform $\mathrm{poly}(\lambda)$-time adversaries.

Definition 3.1 (Pseudorandom Function)**.** *A function $F(\mathsf{sk}, x)$ is called a* pseudorandom function (PRF) *if it satisfies the following.*

1. ***Syntax:*** *F is polynomial-time computable; moreover for any $\lambda \in \mathbb{N}$, key $\mathsf{sk} \in \{0,1\}^\lambda$ and input $x \in \{0,1\}^\lambda$, we have $F(\mathsf{sk}, x) \in \{0,1\}^\lambda$.*
2. ***Pseudorandomness:*** *For any non-uniform polynomial-time algorithm $\mathcal{A}$ that has access to an oracle, the advantage*

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

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

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

3.2 Secret-key Private Information Retrieval

We recall the definition of secret-key PIR (sk-PIR) [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 $x \in \{0,1\}^k$ into an encoded database $\widehat{x} \in \{0,1\}^n$ 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 $o(k)$ bits with the server, without revealing i to the server.

Definition 3.2 (Secret-key PIR)**.** *A* secret-key private information retrieval *protocol (*sk-PIR*) is a tuple of PPT algorithms* $\mathsf{skPIR} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Query}, \mathsf{Ans}, \mathsf{Dec})$ *with the following syntax:*

- $\mathsf{Gen}(1^\lambda, 1^k)$*: The* key generation algorithm *takes a security parameter* λ *and database size* k *and outputs a* secret key sk*. We assume, w.l.o.g., that* sk *contains* λ *and* k*.*
- $\mathsf{Enc}(\mathsf{sk}, x)$*: The* database encoding algorithm *takes a secret key* sk *and a database* $x \in \{0,1\}^k$ *and outputs a database encoding* $\widehat{x} \in \{0,1\}^n$*.*
- $\mathsf{Query}(\mathsf{sk}, i)$*: The* query algorithm *takes a secret key* sk *and an index* $i \in [k]$*, and outputs a PIR-query* $\mathsf{que} \in \{0,1\}^\ell$ *and auxiliary information* $\mathsf{aux} \in \{0,1\}^*$*.*
- $\mathsf{Ans}(\widehat{x}, \mathsf{que})$*: The* answer algorithm *takes a database encoding* $\widehat{x} \in \{0,1\}^n$ *and a PIR-query* que*, and returns a PIR-answer* $\mathsf{ans} \in \{0,1\}^a$*.*
- $\mathsf{Dec}(\mathsf{sk}, i, \mathsf{ans}, \mathsf{aux})$*: The* decoding algorithm *takes a secret key* sk*, an* $i \in [k]$*, 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 Ans *and* Dec *are deterministic. We refer to* n *as the* database encoding size *or* server storage*, and to* k/n *as the* storage rate *of* skPIR*. We say that the storage rate approaches* R *if, for any* $\lambda \in \mathbb{N}$*,*

$$\lim_{k\to\infty} k/n = R.$$

The communication complexity *of* skPIR *consists of the* query length ℓ *and the* answer length a*.*

Correctness. *For any* $\lambda, k \in \mathbb{N}$*,* $\mathsf{sk} \in \mathsf{Gen}(1^\lambda, 1^k)$*,* $x \in \{0,1\}^k$*,* $\widehat{x} \in \mathsf{Enc}(\mathsf{sk}, x)$*,* $i \in [k]$*, and* $(\mathsf{que}, \mathsf{aux}) \leftarrow \mathsf{Query}(\mathsf{sk}, i)$*, we have* $\mathsf{Dec}(\mathsf{sk}, i, \mathsf{Ans}(\widehat{x}, \mathsf{que}), \mathsf{aux}) = x_i$*.*

Security. *We say that* skPIR *is* secure *if for any non-uniform polynomial-time algorithm* $\mathcal{A}$ *and any polynomial* $k = k(\lambda)$*, there exists a negligible function* δ *such that for any* $\lambda \in \mathbb{N}$ *and* $x \in \{0,1\}^k$*, it holds that*

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

where $\mathsf{sk} \leftarrow \mathsf{Gen}(1^\lambda, 1^k)$*,* $\widehat{x} \leftarrow \mathsf{Enc}(\mathsf{sk}, x)$ *and* $\mathsf{Query}'(\mathsf{sk}, \cdot)$ *is an oracle that ignores its input and outputs* $\mathsf{Query}(\mathsf{sk}, 1)$ *(with fresh randomness).*

Double-Efficiency. *We say that* skPIR *is* doubly-efficient *if the runtime complexity of* Ans*, when it is given RAM access to the database encoding* $\widehat{x}$*, is* $o(k)$*.*

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.

3.3 Coding Theory

A *code* C over an alphabet Σ is a subset of codewords in Σ^n. The length of C is n and its dimension is $k = \log_\Sigma |C|$. The rate of a code is defined as $R = k/n$. The minimum (Hamming) distance of a code is defined as $d_H(C) = \min_{\mathbf{c},\mathbf{c}' \in C} d_H(\mathbf{c}, \mathbf{c}')$. A code is often associated with a 1-1 encoding function that maps Σ^k to C.

A code C over a field $\mathbb{F}$ is linear if any linear combination of codewords is itself a codeword. If C is a linear code, it holds that $d_H(C) = \min_{\mathbf{c} \in C} w_H(\mathbf{c})$. We denote the dual of a linear code C by $C^\perp = \{\mathbf{c} \mid \mathbf{c}^\intercal \mathbf{c}' = \mathbf{0},\ \forall \mathbf{c}' \in C\}$, which is also a linear code. A generator matrix for a linear code C is a matrix $\mathbf{G} \in \mathbb{F}^{k \times n}$ satisfying $C = \{\mathbf{x}^\intercal \mathbf{G} \mid \mathbf{x} \in \mathbb{F}_q^k\}$. A parity-check matrix for C is a generator matrix for $C^\perp$.

We recall the definition of Reed-Solomon codes [RS60].

Definition 3.3 (Reed-Solomon Codes)**.** *Let $n, k \in \mathbb{N}$ and let q be a prime power such that $k \leq n \leq q$. The (n,k)-Reed-Solomon code over $\mathbb{F}_q$, with distinct evaluation points $z_1, \ldots, z_n \in \mathbb{F}_q$, is defined as the subset*

$$\mathsf{RS}_q(n, k; z_1, \ldots, z_n) = \{(f(z_1), \ldots, f(z_n)) \ \mid \ f \in \mathbb{F}_q[\alpha]_{<k}\}.$$

We omit $z_1, \ldots, z_n$ from notation when they may be chosen arbitrarily.

A well-known fact is that $d_H(\mathsf{RS}_q(n,k)) = n - k + 1$. Another useful fact concerns the dual distance of Reed-Solomon codes.

Fact 3.1 (Dual Distance of RS Codes)**.** *For any $k \leq n \leq q$, $d_H(\mathsf{RS}(n,k)^\perp) = k + 1$.*

Locally Decodable Codes. A *locally decodable code* (LDC) is a code where any symbol of a message can be recovered by reading a few symbols from the codeword that encodes the message (under a corresponding encoding function). In coding theory, it is often required that decoding succeeds even when reading from a corrupted codeword. In this work, we only require correctness given a noiseless codeword. While local decodability without noise is trivial (as any linear code has a systematic encoding function), we use the LDC syntax to instantiate local decoders with smoothness properties (Definition 4.1) which are essential for cryptographic hardness and are not satisfied by straight-forward decoding.

Definition 3.4 (Locally Decodable Codes: Syntax)**.** *Let $k := k(\lambda)$, $n := n(\lambda)$, $\ell := \ell(\lambda)$ be polynomially bounded functions and $\Sigma := \Sigma(\lambda)$ be an alphabet such that $\log |\Sigma| \leq \mathrm{poly}(\lambda)$.*

An (explicit) *(k, n, ℓ)*-locally decodable code (LDC) *over Σ is a triple of PPT algorithms $\mathsf{LDC} = (\mathsf{E}, \mathsf{Q}, \mathsf{D})$, such that for any $\lambda \in \mathbb{N}$:*

- $\mathsf{E}(1^\lambda, \mathbf{x}) \to \mathbf{c}$ *is a deterministic, injective mapping from Σ^k to Σ^n;*
- $\mathsf{Q}(1^\lambda, i) \to (Q, \mathsf{aux})$ *where $i \in [k]$, $Q \in [n]^\ell$ is a list of ℓ locations and $\mathsf{aux} \in \{0,1\}^*$;*
- $\mathsf{D}(1^\lambda, i, \mathbf{c}_Q, \mathsf{aux}) \to y$, *where $\mathbf{c}_Q \in \Sigma^\ell$ and $y \in \Sigma$.*

We require perfect correctness: for all $\lambda \in \mathbb{N}$, $x \in \Sigma^k$ and $i \in [k]$,

$$\Pr_r[\mathsf{D}(1^\lambda, i, \mathbf{c}_Q, \mathsf{aux}) = \mathbf{x}_i \ \mid \ \mathbf{c} = \mathsf{E}(1^\lambda, \mathbf{x}),\ (Q, \mathsf{aux}) \leftarrow \mathsf{Q}(1^\lambda, i)] = 1.$$

An (efficient) family of (k, n, ℓ)-locally decodable codes (LDC family) *over Σ is defined by $\mathcal{LDC} = (\mathsf{G}, \mathsf{E}, \mathsf{Q}, \mathsf{D})$ where:*

- $\mathsf{G}(1^\lambda) \to \mathsf{id}$ *is a PPT algorithm outputting an LDC identifier* id *(which includes* λ*);*
- *The algorithms* $\mathsf{E}, \mathsf{Q}, \mathsf{D}$ *are defined as above, except for taking* id *instead of* 1^λ*.*

We let $\mathsf{LDC}_{\mathsf{id}}$ *denote the LDC defined by* $(\mathsf{E}(\mathsf{id}, \cdot), \mathsf{Q}(\mathsf{id}, \cdot), \mathsf{D}(\mathsf{id}, \cdot))$ *and require that for every* id*,* $\mathsf{LDC}_{\mathsf{id}}$ *is perfectly correct as above.*

We omit 1^λ *from syntax when it is clear from context.*

All the LDC constructions in this work are in fact constructions of *locally correctable codes (LCCs)*. In LCCs, the decoder's task is to recover a codeword symbol $\mathbf{c}_i$ for $i \in [n]$, rather than a message symbol $\mathbf{x}_i$ for $i \in [k]$, by reading few symbols from $\mathbf{c}$. LCCs are a generalization of LDCs assuming the corresponding encoding function is systematic (which is w.l.o.g. in linear codes). We adhere to the LDC terminology since we use only local decodability for constructing sk-PIR. However, we often define the query and decoding algorithms to take as input $i \in [n]$.

We recall the definition of Reed-Muller codes [Ree54, Mul54] – a generalization of Reed-Solomon codes to the multivariate setting.

Definition 3.5 (Reed-Muller Codes)**.** *Let* $m, d \in \mathbb{N}$ *and let* q *be a prime power. The* m*-variate degree-*d q*-ary Reed-Muller code is defined as the subset*

$$\mathsf{RM}_q(m, d) = \{(f(\mathbf{z}))_{\mathbf{z} \in \mathbb{F}_q^m} \mid f \in \mathbb{F}_q[\alpha_1, \ldots, \alpha_m]_{\leq d}\}.$$

We associate $\mathsf{RM}_q(m, d)$ *with a systematic encoding function that maps any* $\mathbf{x} \in \mathbb{F}_q^k$*, for* $k = \binom{m+d}{m}$*, to* $\mathbf{c} \in \mathsf{RM}_q(m, d)$ *satisfying* $\mathbf{c}_{1,\ldots,k} = \mathbf{x}$*.*

4 Constructions of t-Smooth LDCs

We construct two new families of locally decodable codes, where the queries made by the local decoder are *t-smooth* for a tunable parameter t. The new constructions improve the rate of the standard RM-based construction, in particular allowing super-constant t with constant rate over large enough fields.

Definition 4.1 (Smooth LDC)**.** *Let* LDC *be a* (k, n, ℓ)*-LDC, and let* $t := t(\lambda)$ *be such that* $t < \ell$*. We say that* LDC *is* t-smooth *if for every* $\lambda \in \mathbb{N}$*,* $i \in [k]$*, and* $T \subseteq [\ell]$ *of size* t*, the projected query* Q_T*, for* $Q \leftarrow \mathsf{LDC.Q}(1^\lambda, i)$*, is uniform over* $[n]^t$*. We say that* LDC *is* weakly t-smooth *if instead of requiring* Q_T *to be uniform, we only require that it does not depend on* i*.*

Recall the m-variate, degree-d Reed-Muller codes $\mathsf{RM}_q(m, d)$ (Definition 3.5), which are well-known to be t-smooth locally decodable whenever $dt + 1 < q$. To recover the i^{th} symbol in a codeword $\mathbf{c} \in \mathsf{RM}_q(m, d)$ (for $i \in \mathbb{F}_q^m$), the local decoder reads the locations defined by degree-t parametric curve ψ, namely the locations $Q = \{\psi(z) = (\psi_1(z), \ldots, \psi_m(z)) \mid z \in \mathbb{F}\}$, where $\psi_j \in \mathbb{F}_q[\alpha]_{\leq t}$ are sampled uniformly at random subject to $\psi(0) = i$. In fact, it is sufficient for the local decoder to read any subset of size $dt + 1$ points on the curve, since such a subset uniquely determines the values on the rest of the points by the polynomial identity lemma (Schwartz-Zippel): $\mathbf{c}$ is the truth table of some degree-d polynomial $f \in \mathbb{F}_q[\alpha_1, \ldots, \alpha_m]_{\leq d}$ (Definition 3.5), therefore $f \circ \psi$ is a polynomial of degree at most dt. This redundancy is used in the coding-theoretic context to obtain a local decoder that tolerates a bounded amount of corruptions by reading the entire curve. In our context, we use it to produce entropy in the locations queried by the local decoder, which we define to sample a random subset of the curve defined by random evaluation points $z_1, \ldots, z_{dt+1}$. Smoothness follows by the fact that any t points in a uniformly random degree-t curve are uniformly random given $\psi(0)$.

Proposition 4.1 (Smooth Local Decodability of RM via Curves)**.** *Let $m := m(\lambda)$ and $d := d(\lambda)$ be polynomially bounded and let $q := q(\lambda)$ be a polynomially bounded prime power. For any $t := t(\lambda)$ such that $dt + 1 < q$, there exists a t-smooth (k, n, ℓ)-LDC with $k = \binom{m+d}{m}$, $n = q^m$ and $\ell = dt + 1$.*

The Reed-Muller LDC from Proposition 4.1, which we denote by $\mathsf{LDC}_{\mathsf{RM\text{-}RS}} = (\mathsf{E}_{\mathsf{RM}}, \mathsf{Q}_{\mathsf{RS}}, \mathsf{D}_{\mathsf{RS}})$, consists of E_{RM} that computes the systematic $\mathsf{RM}_q(m, d)$ encoding (Definition 3.5), and Q_{RS} and D_{RS} that are defined in Fig. 3. We use the subscript RS since the decoder's query set consists of points on a low-degree curve, which may be viewed as an m-tuple of Reed-Solomon codewords (Definition 3.3). In fact, as we will see next, curve-decoding of RM is possible due to certain algebraic properties underlying RS codes.

t-Smooth Local Decoding of $\mathsf{RM}_q(m, d)$ via Curves:

- $\mathsf{Q}_{\mathsf{RS}}(i \in \mathbb{F}_q^m)$:
  1. Sample a parametric curve $\psi = (\psi_1, \ldots, \psi_m)$ where $\psi_j \in \mathbb{F}_q[\alpha]_{\leq t}$ are uniformly random subject to $\psi(0) = i$.
  2. Sample $\ell = dt + 1$ distinct uniformly random elements $z_1, \ldots, z_\ell \in \mathbb{F}_q^*$.
  3. Output $Q = (\psi(z_1), \ldots, \psi(z_\ell))$ and $\mathsf{aux} = (z_1, \ldots, z_\ell)$.
- $\mathsf{D}_{\mathsf{RS}}(i, \mathbf{c}_Q, \mathsf{aux})$:
  1. Let $f \in \mathbb{F}_q[\alpha]_{\leq dt}$ be the unique polynomial that satisfies $\mathbf{c}_Q = (f(z_1), \ldots, f(z_\ell))$.
  2. Output $f(0)$.

Figure 3: Smooth Local Decoding of Reed-Muller Codes via Low-degree Curves (Proposition 4.1).

The rate of an $\mathsf{RM}_q(m, d)$ code is $R = \binom{m+d}{m}/q^m \geq \frac{1}{m!} \cdot \left(\frac{d}{q}\right)^m$. By the constraints of Proposition 4.1, the rate of RM codes vanishes with the smoothness parameter t given by curve-based decoding.

Corollary 4.1 ($\mathsf{LDC}_{\mathsf{RM\text{-}RS}}$ asymptotic rate)**.** *For any polynomially bounded $m := m(\lambda)$ and $t := t(\lambda)$, there exist monotone polynomials $q := q(\lambda)$ and $k := k(\lambda)$, for which there is a t-smooth (k, n, ℓ)-LDC over $\mathbb{F}_q$ with rate approaching $1/(m! \cdot t^m)$ and locality $\ell = O(mt \cdot k^{1/m})$.*

4.1 Smoothly Decoding RM Codes via Concatenated RS Codes

In our first new construction, we propose a new decoder for RM codes that allows to increase the smoothness parameter t for any given rate. Unlike the curve-based decoder, however, the new decoder achieves only a weak notion of smoothness (Definition 4.1).

Theorem 4.1 (Weakly Smooth Local Decodability of RM via Concatenated RS)**.** *Let $m := m(\lambda)$, $d := d(\lambda)$ and $e := e(\lambda)$ be polynomially bounded and let $q := q(\lambda)$ be a polynomially bounded prime power such that $q > d(e-1)$. For any $t := t(\lambda)$ such that $dt+1 < q^e$, there exists a weakly t-smooth (k, n, ℓ)-LDC with $k = \binom{m+d}{m}$, $n = q^m$ and $\ell = (d(e-1)+1)(dt+1)$.*

Notice that a special case of the above, with $e = 1$, gives the standard local decoding of RM codes (Proposition 4.1).

Theorem 4.1 allows us to obtain (weak) t-smoothness with rate that decays as slow as a polynomial in $e \approx \log_q(t)$. Setting e to be a small constant, e.g. 2, and setting $t = q = d + 1$, we get an LDC with rate that does not decay with t.

Weakly t-Smooth Local Decoding of $\mathsf{RM}_q(m,d)$ via Concatenated RS:

Fix an irreducible degree-e polynomial $p \in \mathbb{F}_q[\alpha]$.
For $w_1, \ldots, w_r \in \mathbb{F}_q$:

$\varphi_{w_1,\ldots,w_r} : \mathbb{F}_{q^e} \to \mathbb{F}_q^r$ maps any polynomial $f \in \mathbb{F}_q[\alpha]/\langle p \rangle \cong \mathbb{F}_{q^e}$ to $(f(w_1), \ldots, f(w_r)) \in \mathbb{F}_q^r$.

$\varphi^{-1}_{w_1,\ldots,w_r} : \mathbb{F}_q^r \to \mathbb{F}_{q^e}$ maps any $\mathbf{y} \in \mathbb{F}_q^r$, to $f \mod p \in \mathbb{F}_q[\alpha]/\langle p \rangle \cong \mathbb{F}_{q^e}$, where f is the unique polynomial $f \in \mathbb{F}_q[\alpha]_{<q}$ satisfying $\mathbf{y} = (f(w_1), \ldots, f(w_r))$.

- $\mathsf{Q}_{\mathsf{Conc}}(i \in \mathbb{F}_q^m)$:

1. Sample a parametric curve $\psi = (\psi_1, \ldots, \psi_m)$ where $\psi_j \in \mathbb{F}_{q^e}[\alpha]_{\leq t}$ are uniformly random subject to $\psi(0) = i$.
2. Sample $s = dt + 1$ distinct uniformly random elements $z_1, \ldots, z_s \in \mathbb{F}_{q^e}^*$.
3. For $j = 1, \ldots, s$, sample $r = d(e-1)+1$ distinct uniformly random elements $w_{j,1}, \ldots, w_{j,r} \in \mathbb{F}_q$.
4. For $h = 1, ..., m$, let $\mathbf{v}_h = \left(\varphi_{w_{j,1},\ldots,w_{j,r}}(\psi_h(z_j))\right)_{j=1,\ldots,s} \in \mathbb{F}_q^\ell$, where $\ell = s \cdot r$, and denote $\mathbf{v}_h = (v_{h,1}, \ldots, v_{h,\ell})$.
5. Output $Q = ((v_{1,1}, \ldots, v_{m,1}), \ldots, (v_{1,\ell}, \ldots, v_{m,\ell}))$ and $\mathsf{aux} = ((z_1, \ldots, z_\ell), (w_{j,1}, \ldots, w_{j,r})_{j=1,\ldots,s})$.

- $\mathsf{D}_{\mathsf{Conc}}(i, \mathbf{c}_Q \in \mathbb{F}_q^\ell, \mathsf{aux})$:

1. Write $\mathbf{c}_Q = (\mathbf{y}_1, \ldots, \mathbf{y}_s)$ for $\mathbf{y}_j \in \mathbb{F}_q^r$ and let $u_j = \varphi^{-1}_{w_{j,1},\ldots,w_{j,r}}(\mathbf{y}_j) \in \mathbb{F}_{q^e}$.
2. Let $f \in \mathbb{F}_{q^e}[\alpha]_{\leq dt}$ be the unique polynomial that satisfies $u_j = f(z_j)$.
3. Output $f(0)$.

Figure 4: Weakly Smooth Local Decoding of Reed-Muller Codes via Concatenated-RS Curves (Theorem 4.1).

Corollary 4.2 ($\mathsf{LDC}_{\mathsf{RM\text{-}Conc}}$ asymptotic rate)**.** *For any polynomially bounded $m := m(\lambda)$ and $t := t(\lambda)$, there exist monotone polynomials $q := q(\lambda)$ and $k := k(\lambda)$ for which there is a weakly t-smooth (k, n, ℓ)-LDC over $\mathbb{F}_q$ with rate k/n approaching $1/m!$ and locality $\ell = O(m \cdot k^{3/m})$.*

For a higher confidence in security in the sk-DEPIR application, one may opt for a smaller value of d/larger value of q, in exchange for a worse rate; see further discussion in Section 6.1.2. The rate remains constant, however, as long as m is constant and $d = \Omega(q)$.

The local decoder of Theorem 4.1 is presented in Fig. 4 and we denote it by $\mathsf{LDC}_{\mathsf{RM\text{-}Conc}} = (\mathsf{E}_{\mathsf{RM}}, \mathsf{Q}_{\mathsf{Conc}}, \mathsf{D}_{\mathsf{Conc}})$ the corresponding LDC. The decoder uses "concatenated" low-degree curves for decoding, namely degree-t curves over an extension field of $\mathbb{F}_q$, where every point on the curve is represented by a smaller curve, of degree $e \ll t$, over the base field $\mathbb{F}_q$. Just as curve decoding corresponds to tuples of RS codewords and relies on algebraic properties of RS, the new decoder outputs queries that correspond to codewords of *concatenated Reed-Solomon codes* and its correctness emerges from similar properties of these codes.

We prove Theorem 4.1 by a generic framework for RM local decoding, that unifies both the standard curve-based decoder (Proposition 4.1) and the new decoder.

We begin by identifying a useful property in the code used for generating the decoder's queries.

Definition 4.2 (Code Multiplication)**.** *For a code A over a field $\mathbb{F}$, define its d-wise multiplication as*

$$A^{*d} = \mathbf{Span}_{\mathbb{F}_q}\{\mathbf{a}_1 * \mathbf{a}_2 * \cdots * \mathbf{a}_d \mid \mathbf{a}_i \in A\},$$

where $*$ denotes point-wise product.

Lemma 4.1 (Smooth Local Decodability of RM, Generically)**.** *Let $\ell := \ell(\lambda)$, $t := t(\lambda)$, $m := m(\lambda)$ and $d := d(\lambda)$ be polynomially bounded and let $q := q(\lambda)$ be a prime power such that $\log q = \mathrm{poly}(\lambda)$. Assume there exists a linear code $A := A(\lambda)$ of length $\ell + 1$ over $\mathbb{F}_q$, with a generator matrix uniformly computable in time $\mathrm{poly}(\lambda)$, where $d_H(A^{*d}) \geq 2$ and $d_H(A^{\perp}) \geq t + 2$. Then, the Reed-Muller code $\mathsf{RM}_q(m, d)$ admits a t-smooth (k, n, ℓ)-LDC with $k = \binom{m+d}{m}$ and $n = q^m$.*

Proof. The locally decodable code consists of the Reed-Muller systematic encoding function as defined in Definition 3.5. The query algorithm $\mathsf{Q}(1^\lambda, i \in \mathbb{F}_q^m)$ picks m codewords $\mathbf{a}_1, \ldots, \mathbf{a}_m \leftarrow A$ uniformly at random, where $\mathbf{a}_h = (a_{h,0}, a_{h,1}, \ldots, a_{h,\ell})$, conditioned on $i = (a_{1,0}, a_{2,0}, \ldots, a_{m,0})$. (This can be efficiently performed given a generator matrix for A.) The algorithm then outputs a query set $Q = (\mathbf{b}_1, \ldots, \mathbf{b}_\ell)$ where $\mathbf{b}_j = (a_{1,j}, a_{2,j}, \ldots, a_{m,j})$.

Since $d_H(A^{\perp}) \geq t + 2$, the coordinates $(\mathbf{b}_{j_1}, \ldots, \mathbf{b}_{j_t})$ are jointly random for any $1 \leq j_1 < \cdots < j_t \leq r$, even given i, hence the local decoder is t-smooth.

For correctness, assume that we want to locally decode the codeword $\mathbf{c} = (f(\mathbf{z}))_{\mathbf{z} \in \mathbb{F}_q^m} \in \mathsf{RM}_q(m, d)$ at position $i \in \mathbb{F}_q^m$. As $(f(i), f(\mathbf{b}_1), \ldots, f(\mathbf{b}_\ell))$ is a codeword of A^{*d} (f is a linear combination of monomials, each a point-wise product of size at most d) and $d_H(A^{*d}) \geq 2$, $\mathbf{c}_i = f(i)$ is uniquely determined by $\mathbf{c}_Q = (f(\mathbf{b}_1), \ldots, f(\mathbf{b}_\ell))$. Further, by the linearity of A^{*d}, $\mathbf{c}_i$ is simply a linear combination over $\mathbf{c}_Q$ which depends solely on the code A and the locations $z_1, \ldots, z_\ell$ and defines an efficient decoding algorithm D. □

Proposition 4.1 is a corollary of Lemma 4.1, by plugging in as A the Reed-Solomon code with length $\ell + 1$, where $\ell = dt + 1$, and dimension $t + 1$, namely $\mathsf{RS}_q(\ell + 1, t + 1; z_1, \ldots, z_{\ell+1})$ with any choice of evaluation points $z_1, \ldots, z_{\ell+1} \in \mathbb{F}_q$.

Lemma 4.2. *For any $d(k - 1) + 2 \leq n \leq q$, $d_H(\mathsf{RS}_q(n, k)^{*d}) \geq 2$.*

Proof. The lemma follows from the observation that $\mathsf{RS}_q(n, k)^{*d} = \mathsf{RS}_q(n, d(k - 1) + 1)$ and the fact that $d_H(\mathsf{RS}_q(n, d(k - 1) + 1)) = n - d(k - 1)$. □

By Lemma 4.2, $A = \mathsf{RS}_q(\ell + 1, t + 1)$ satisfies $d_H(A^{*d}) \geq 2$ when $\ell = dt + 1$. Further, it holds that $d_H(A^{\perp}) = t + 2$ (Fact 3.1). Overall, A satisfies the conditions of Lemma 4.1 and we may indeed use it to obtain Proposition 4.1 as a corollary. The corresponding smooth local decoder reads the locations defined by random RS codewords, namely by a random low-degree parametric curve.

Notice, however, that the RS-based local decoder of Fig. 3 samples uniformly random evaluation points $z_1, \ldots, z_\ell$ which, together with $z_0 = 0$, define a random instance of $\mathsf{RS}_q(\ell + 1, t + 1)$ that is used to derive the queries. In a sense, one may think of that decoder as invoking a random decoder from a family of decoders, each defined by a different $\mathsf{RS}_q(\ell + 1, t + 1)$ instance, i.e. a different choice of evaluation points.

In general, we may replace the code A in Lemma 4.1 with a distribution over codes $\mathcal{A}$ that satisfy the desired properties. The query algorithm Q samples $A \leftarrow \mathcal{A}$ and uses it for its queries. The choice of A is passed from Q to the decoding algorithm D via the auxiliary input aux (as done in Fig. 3).

Remark 4.1. *The implication of Lemma 4.1 holds assuming an efficiently samplable distribution over linear codes $A \leftarrow \mathcal{A}$ that satisfies the conditions in the lemma with probability 1.*

The decoder from Lemma 4.1 requires that the LDC and the code A are over the same alphabet, limiting our design space. We extend the above framework by allowing A to be defined over any extension field of $\mathbb{F}_q$. To map any m-tuple of codewords $\mathbf{a}_1, \ldots, \mathbf{a}_m \in A$ over the extension field

to a query set over the coordinate space $\mathbb{F}_q^m$, we use *multiplication-friendly pairs* of mappings that preserve the redundant nature of the multiplication code A^{*d} by a correspondence between multiplication in the extension field and point-wise product of base field coordinates.

Definition 4.3 (Multiplication-Friendly Pairs)**.** *A pair (φ, φ^{-1}) is called a $(d, e, r)_q$-multiplication friendly pair if φ is an $\mathbb{F}_q$-linear map from $\mathbb{F}_{q^e}$ to $\mathbb{F}_q^r$ and φ^{-1} is an $\mathbb{F}_q$-linear map from $\mathbb{F}_q^r$ to $\mathbb{F}_{q^e}$ such that $\varphi(1) = (1, \ldots, 1)$ and $\varphi^{-1}(\varphi(\alpha_1) * \cdots * \varphi(\alpha_d)) = \prod_i \alpha_i$ for all $\alpha_i \in \mathbb{F}_{q^e}$.*

Note that in a multiplication friendly pair, φ is necessarily injective.

Since applying a multiplication-friendly mapping to the smooth query sets defined by A does not generally preserve smoothness, we obtain a decoder that is only weakly smooth.

Lemma 4.3 (Weakly Smooth Local Decodability of RM Codes, Generically)**.** *Let $e := e(\lambda)$, $s := s(\lambda)$, $r := r(\lambda)$, $t := t(\lambda)$, $m := m(\lambda)$ and $d := d(\lambda)$ be polynomially bounded and let $q := q(\lambda)$ be a prime power such that $\log q = \mathrm{poly}(\lambda)$. Assume there exists:*

1. *a linear code $A := A(\lambda)$ of length $s+1$ over $\mathbb{F}_{q^e}$, with a generator matrix uniformly computable in time $\mathrm{poly}(\lambda)$, where $d_H(A^{*d}) \geq 2$ and $d_H(A^{\perp}) \geq t + 2$,*
2. *a $(d, e, r)_q$-multiplication-friendly pair $(\varphi, \varphi^{-1}) := (\varphi(\lambda), \varphi^{-1}(\lambda))$ where both mappings are uniformly computable in time $\mathrm{poly}(\lambda)$.*

Then, there exists a weakly t-smooth (k, n, ℓ)-LDC with $k = \binom{m+d}{m}$, $n = q^m$ and $\ell = s \cdot r$.

Proof. Once again, we use the systematic encoding of RM codes (Definition 3.5). The query algorithm $\mathsf{Q}(1^\lambda, i \in \mathbb{F}_q^m)$ picks m codewords $\mathbf{a}_1, \ldots, \mathbf{a}_m \leftarrow A$ uniformly at random, where $\mathbf{a}_h = (a_{h,0}, a_{h,1}, \ldots, a_{h,s})$, conditioned on $i = (a_{1,0}, a_{2,0}, \ldots, a_{m,0})$, similarly to Lemma 4.1. However, here the codewords are over $\mathbb{F}_{q^e}$ and they cannot be directly used to define coordinates in the Reed-Muller coordinate space $\mathbb{F}_q^m$. To derive the coordinates, we use the mapping φ.

To make things more concrete, consider the natural generalization of the mappings φ and φ^{-1} to vectors, where they are applied in a point-wise and, respectively, block-wise manner (to consecutive blocks of length r).

The algorithm Q computes $\mathbf{v}_h = \varphi(a_{h,1}, \ldots, a_{h,s}) \in \mathbb{F}_q^\ell$ and writing it as $\mathbf{v}_h = (v_{h,1}, \ldots, v_{h,\ell})$, outputs the query set $Q = (\mathbf{q}_1, \ldots, \mathbf{q}_\ell)$, where $\mathbf{q}_j = (v_{1,j}, \ldots, v_{m,j})$.

Precisely as in the smoothness argument behind Lemma 4.1, any $(a_{1,j_1}, \ldots, a_{m,j_1}), \ldots, (a_{1,j_t}, \ldots, a_{m,j_t})$, for distinct $j_1, \ldots, j_t$, are jointly uniform given i since $d_H(A^{\perp}) \geq t + 2$. Since any coordinate in Q is determined by $(a_{1,j}, \ldots, a_{m,j})$ for some j, we immediately obtain weak t-smoothness.

Define

$$B = \{(a_0, \varphi(a_1, \ldots, a_s)) \mid (a_0, \ldots, a_s) \in A\} \subset \mathbb{F}_{q^e} \times \mathbb{F}_q^\ell. \tag{1}$$

By the multiplication-friendliness of (φ, φ^{-1}), for any $(b_0, \mathbf{b}) \in B^{*d}$ it holds that $(b_0, \varphi^{-1}(\mathbf{b})) \in A^{*d}$. In particular, $d_H(B^{*d}) \geq d_H(A^{*d}) \geq 2$.

Let $\mathbf{c} = (f(\mathbf{z}))_{z \in \mathbb{F}_q^m} \in \mathsf{RM}_q(m, d)$. Once again, $(\mathbf{c}_i, \mathbf{c}_Q) = (f(i), f(\mathbf{q}_1), \ldots, f(\mathbf{q}_\ell))$ is a codeword of B^{*d} and therefore the decoding algorithm $\mathsf{D}(1^\lambda, i, \mathbf{c}_Q)$ can recover $\mathbf{c}_i$ by computing a fixed linear combination over $\mathbf{c}_Q$ (defined by the code A and the mapping φ^{-1}). □

Note that Lemma 4.1 is a special case of Lemma 4.3 with $e = 1$ and (φ, φ^{-1}) that are the identity mappings.

We instantiate Lemma 4.3 with A that is the Reed-Solomon code over $\mathbb{F}_{q^e}$ with length $s + 1$ and dimension $t + 1$, i.e. $\mathsf{RS}_{q^e}(s + 1, t + 1)$ (Definition 3.3), and the pair of mappings (φ, φ^{-1}) from Fig. 4 which is $(d, e, r)_q$-multiplication-friendly pair.

Lemma 4.4. *For any $d, e, q \in \mathbb{N}$ such that $r := d(e-1)+1 \leq q$ and any distinct $w_1, \ldots, w_r \in \mathbb{F}_q$, the pair $(\varphi_{w_1,\ldots,w_r}, \varphi^{-1}_{w_1,\ldots,w_r})$ described in Fig. 4 is a $(d, e, r)_q$-multiplication-friendly pair.*

Proof. The lemma follows by the fact that for any $\alpha_1, \ldots, \alpha_d \in \mathbb{F}_q[\beta]_{<e} \subset \mathbb{F}_q[\beta]/\langle p \rangle$ the vector $\mathbf{y} = \varphi(\alpha_1) * \cdots * \varphi(\alpha_d) \in \mathbb{F}_q^r$ contains the values of the product polynomial $\pi = \prod_j \alpha_j$ at $w_1, \ldots, w_r$ which is of degree at most $d(e-1) < q$. Hence, $\varphi^{-1}(\mathbf{y})$ returns $\prod_j \alpha_j \mod p$. □

Theorem 4.1 then follows as a corollary of Lemma 4.3, via Fact 3.1 and Lemmas 4.2 to 4.4, by setting $r = d(e-1)+1$ and $s = dt+1$.

Recall the local decoder underlying Theorem 4.1 derives its queries from codewords of the code B (defined in Eq. (1)), which is essentially the concatenation of the code A with the mapping φ. Since in our instantiation both A and φ are variants of Reed-Solomon codes, we obtain a local decoder that is based on concatenated RS codes.

The local decoder we define in Fig. 4 is however slightly different than what we describe here, as its queries are defined by the values of low-degree polynomials (or curves) at randomly sampled subsets of evaluation points (denoted $z_1, \ldots, z_s$ and $w_{j,1}, \ldots, w_{j,r}$). Again, one can think of this as sampling a random decoder from a family of decoders, each defined w.r.t. a different instance of RS code $A = \mathsf{RS}(s+1, t+1; 0, z_1, \ldots, z_s)$ and a different RS-based multiplication-friendly pair $(\varphi_{w_1,\ldots,w_r}, \varphi^{-1}_{w_1,\ldots,w_r})$.

Remark 4.2. *The implication of Lemma 4.3 holds assuming an efficiently samplable distribution over linear codes $A \leftarrow \mathcal{A}$ and an efficiently samplable distribution over multiplication-friendly pairs $(\varphi, \varphi^{-1}) \leftarrow \Phi$ both satisfying the conditions in the lemma with probability 1.*

Lastly, we note that in order to obtain weak smoothness we may relax the condition $d_H(A^\perp) \geq t+2$ in Lemma 4.3 to only require that the first symbol of a codeword is independent in any other t symbols. This is weaker than requiring that any t symbols in a random codeword distribute uniformly at random, and is equivalent to the condition that $(1, 0^t) \in A|_{\{1\}\cup T}$ for any $T \subset \{2, \ldots, n\}$ of size t. (Recall $A|_{\{1\}\cup T}$ denotes the restriction of A to the coordinates in $\{1\} \cup T$.)

4.2 Curve-Lifted RS Codes

Our second new construction of LDCs achieves rate that approaches 1 with super-constant smoothness in certain parameter regimes.

Theorem 4.2 (High-Rate Smooth LDC via Curve Decoding)**.** *Let $m := m(\lambda) \geq 2$ be polynomially bounded and let $q := q(\lambda)$ be a polynomially bounded prime power. For any $t := t(\lambda) < q-1$, there exists a t-smooth $(k, n, q-1)$-LDC with $n = q^m$ and $k = R \cdot n$, where the rate is*

$$R \geq 1 - tm \cdot 2^{4tm^2} \cdot q^{-\varepsilon(m,t)},$$

for $\varepsilon(m,t) \geq 2^{-tm^2}/(tm \cdot \ln 2)$.

Given sufficiently small m (possibly super-constant), we may set the smoothness parameter to be a sufficiently small super-constant and q a sufficiently large polynomial to obtain a rate that asymptotically approaches 1. For instance, setting t so that $tm^2 = 0.9 \log\log(\lambda)$ gives $\varepsilon(m,t) = \widetilde{\Omega}(\log^{-0.9}(\lambda))$ and the following corollary.

Corollary 4.3 ($\mathsf{LDC}_{\mathsf{Lift\text{-}RS}}$ Asymptotic Rate)**.** *For any integer $m := m(\lambda) \geq 2$ and any $t := t(\lambda) \leq (0.9/m^2) \cdot \log\log\lambda$, there exist polynomially bounded monotone functions $q := q(\lambda)$ and $k := k(\lambda)$ for which there is a t-smooth (k, n, ℓ)-LDC over $\mathbb{F}_q$ with rate approaching 1 and locality $\ell = O(k^{1/m})$.*

Our construction is based on curve-lifted Reed-Solomon codes. These are codes where any restriction of a codeword to a low-degree curve constitutes a Reed-Solomon codeword.

Definition 4.4 (Curve-Lifted Reed-Solomon)**.** *Let $m, k, t \in \mathbb{N}$ and let q be a prime power. The* curve-lifted Reed-Solomon *code* $\mathsf{Lift\text{-}RS}_q(m, \ell, t)$ *is the set of all codewords* $\mathbf{c} : \mathbb{F}_q^m \to \mathbb{F}_q$ *such that for any degree-t parametric curve $\psi = (\psi_1, \ldots, \psi_m)$ (where $\psi_j \in \mathbb{F}_q[\alpha]_{\leq t}$), it holds that*

$$(\mathbf{c}(\psi(z)))_{z \in \mathbb{F}_q} \in \mathsf{RS}_q(q, \ell).$$

The t-smooth local decodability of lifted RS codes is implied directly by definition; just as in Fig. 3, the decoder reads ℓ points from a degree-t curve and interpolates the value of the curve at 0 (we replace the choice of ℓ in Fig. 3 with the dimension of the underlying RS).

We restrict our attention to the case of $\ell = q - 1$, as hinted by Theorem 4.2. This is where the minimum distance between codewords when restricted to curves is 2, namely the smallest distance that allows local decoding via curves. By default, we denote $\mathsf{Lift\text{-}RS}_q(m, t) := \mathsf{Lift\text{-}RS}_q(m, q-1, t)$.

Known Asymptotics for the Rate of Curve-Lifted RS codes. The special case $t = 1$ reduces to restricting on affine lines: degree-$\leq$1 parametric curves are exactly affine lines in $\mathbb{F}_q^m$, so $\mathsf{Lift\text{-}RS}_q(m, 1)$ coincides with the usual m-variate *(line-)lifted Reed-Solomon* code obtained by lifting the base code $\mathsf{RS}(q, q-1)$ along all lines, as introduced by Guo-Kopparty-Sudan [GKS13] (see also [GK14]).

For line-lifted Reed-Solomon codes more generally (base dimension $k = q - r$), the asymptotic rate for fixed m and growing field size q is understood. The original work [GKS13], showed that these codes asymptotically achieved rate 1. Furthermore, Holzbaur-Polyanskaya-Polyanskii-Vorobyev [HPPV20] gave a more precise bounds on the rate achieved by line lifted codes.

Several works also study more general curve-based variants of lifting. Lavauzelle-Nardi define an *η-lifting* of Reed-Solomon codes in the bivariate setting by requiring the RS constraint on all "η-lines" (graphs of degree-η polynomials), and prove that for fixed η and fixed $\alpha \geq 2$, the rate of the obtained code approaches 1 as $q = p^e$ with $e \to \infty$ (Theorem 5.11 in [LN19]). Liu-Holzbaur-Polyanskii-Puchinger-Wachter-Zeh introduce *quadratic-curve-lifted* RS codes (bivariate, quadratic curves) and derive upper/lower bounds on the dimension; they also obtain an asymptotic rate expression $1 - \Theta((q/r)^{-0.2284})$ in terms of the local redundancy r [LHP+21].

To the best of our knowledge, for the specific multivariate family $\mathsf{Lift\text{-}RS}_q(m, t)$ that quantifies over *all* degree-$\leq t$ parametric curves in $\mathbb{F}_q^m$ (for $m \geq 3$ and $t > 1$), the literature does not currently provide a closed-form asymptotic rate as a function of m, t. One of the contributions of this work is to compute a (somewhat loose) lower bound on the rate of the code as a function of $q = 2^s$, m and t. These bounds hold good for even non-constant choices of s, m and t.

Section Organization. In what follows, we characterize the code $\mathsf{Lift\text{-}RS}_q(m, t)$ by a basis of "good monomials" and show how to find these monomials algorithmically. This aspect of the construction is well-known from prior works, we recall them below for sake of completeness. In Section 4.2.2, we bound the asymptotic rate of the code by bounding the dimension of the basis. Unlike prior works, this bound gives an expression as a function of q, m, t. We additionally provide concrete estimates for the rate in Section 4.2.3 by empirical Monte-Carlo estimations of the number of good monomials in relevant parameter regimes. This might be useful for estimating practicality of our schemes. Lastly, in Section 4.2.4, we show a systematic encoding algorithm for lifted RS codes that runs in time quasi-linear in k.

4.2.1 Monomial Basis for Lifted Reed-Solomon

We say that a linear code $C \subseteq \{\mathbb{F}_q^m \to \mathbb{F}_q\}$ is *affine invariant* if for any $\mathbf{c} \in C$ and any affine function $\varphi : \mathbb{F}_q^m \to \mathbb{F}_q^m$, it holds that $\mathbf{c} \circ \varphi \in C$.

Claim 4.1 (Affine Invariance)**.** $\mathsf{Lift\text{-}RS}_q(m,t)$ *is an* $\mathbb{F}_q$*-linear, affine-invariant code.*

Proof. Linearity is immediate from linearity of the RS constraint over any curve. For affine invariance, fix $A \in \mathrm{GL}_m(\mathbb{F}_q)$ and $b \in \mathbb{F}_q^m$. Let $\mathbf{c} \in \mathsf{Lift\text{-}RS}_q(m,t)$ and define $\mathbf{c}'$ by $\mathbf{c}'(z) = \mathbf{c}(Az+b)$. For any degree-$\leq t$ curve ψ, the composition $\psi'(z) = A\psi(z) + b$ is also a degree-$\leq t$ curve. It follows, then, that $\mathbf{c}'(\varphi(z)) = \mathbf{c}(\varphi'(z)) \in \mathsf{RS}(q, q-1)$. □

Every function $f : \mathbb{F}_q^m \to \mathbb{F}_q$ has a unique representation as a polynomial $p \in \mathbb{F}_q[\alpha_1, \ldots, \alpha_m]_{\leq m(q-1)}$, obtained by reducing modulo $\alpha_1^q - \alpha_1, \ldots, \alpha_m^q - \alpha_m$. In the following, we work with polynomials in the quotient ring $\mathcal{R} = \mathbb{F}_q[\alpha_1, \ldots, \alpha_m]/\langle \alpha_1^q - \alpha_1, \ldots, \alpha_m^q - \alpha_m \rangle$ and, consequently, consider polynomials to be equivalent if they are in the same equivalent class induced by $\mathcal{R}$. We thus consider a notion of reduced degree and assume $\deg(p) \leq m(q-1)$ for any m-variate polynomial p over $\mathbb{F}_q$

The following lemma is standard in the theory of affine-invariant codes; see, e.g., [HRS13, Lemma 3.1].

Lemma 4.5 (Monomial Extraction)**.** *Let* $C \subseteq \{\mathbb{F}_q^m \to \mathbb{F}_q\}$ *be an* $\mathbb{F}_q$*-linear affine-invariant code. Let* $\mathrm{Supp}(C)$ *be the set of monomials (with exponents in* $\{0, \ldots, q-1\}^m$*) that appear with nonzero coefficient in the unique polynomial representation of some codeword. Then* $C = \mathbf{Span}_{\mathbb{F}_q}(\mathrm{supp}(C))$*. In particular,* C *has a monomial basis.*

By Lemma 4.5 and Claim 4.1, then, the lifted RS code is characterized by a basis of monomials that span all its codewords.

Corollary 4.4 (Good Monomials)**.** *There exists a subset* $\mathcal{G} \subseteq \{0, \ldots, q-1\}^m$ *such that*

$$\mathsf{Lift\text{-}RS}_q(m,t) = \mathbf{Span}_{\mathbb{F}_q}\left\{\alpha_1^{e_1} \cdots \alpha_m^{e_m} : (e_1, \ldots, e_m) \in \mathcal{G}\right\}.$$

We call $(e_1, \ldots, e_m) \in \{0, \ldots, q-1\}^m$ good *if its monomial belongs to* $\mathsf{Lift\text{-}RS}_q(m,t)$*.*

Computing the Good Monomials. Having characterized the code $\mathsf{Lift\text{-}RS}_q(m,t)$ by a basis of good monomials, we next show how to algorithmically find such a basis by enumerating over all good monomials.

From now on, the analysis concerns the case where $q = 2^s$ for some $s \in \mathbb{N}$, namely where the characteristic of $\mathbb{F}_q$ is 2.

For any $e = (e_1, \ldots, e_m) \in \{0, \ldots, q-1\}^m$, we denote the corresponding monomial by

$$M_e(\alpha) := \prod_{i=1}^{m} \alpha_i^{e_i}.$$

To understand when M_e is a good monomial, we look at its restriction to the points of a degree-t parametric curve $\psi = (\psi_1, \ldots, \psi_m)$ where $\psi_i(\alpha) = \sum_{j=0}^{t} c_{i,j}\alpha^j$. We want to see when this restriction lies in $\mathsf{RS}_q(q, q-1)$, on par with the definition of the lifted RS code $\mathsf{Lift\text{-}RS}_q(m,t)$.

It holds that

$$M_e(\psi(\alpha)) = \prod_{i=1}^{m} \psi_i(\alpha)^{e_i}.$$

Write each exponent in binary:

$$e_i = \sum_{k=0}^{s-1} e_{i,k} 2^k, \qquad e_{i,k} \in \{0,1\}.$$

By Frobenius in characteristic two,

$$\psi_i(\alpha)^{2^k} = \Big(\sum_{j=0}^{t} c_{i,j}\alpha^j\Big)^{2^k} = \sum_{j=0}^{t} c_{i,j}^{2^k}\alpha^{j2^k}.$$

Hence

$$\psi_i(\alpha)^{e_i} = \prod_{k:e_{i,k}=1} \Big(\sum_{j=0}^{t} c_{i,j}^{2^k}\alpha^{j2^k}\Big).$$

Define the bit-weight of e at bit-position k as

$$\mathrm{wt}_k(e) := \sum_{i=1}^{m} e_{i,k} \in \{0,1,\ldots,m\}.$$

At bit-position k, the total number of factors of the form $\sum_{j=0}^{t}(\cdots)\alpha^{j2^k}$ equals $\mathrm{wt}_k(e)$. Expanding the product, we may choose for each such factor a degree contribution $j \in \{0,\ldots,t\}$, so the net contribution at bit k is some

$$u_k \in \{0,1,\ldots,t\cdot \mathrm{wt}_k(e)\}.$$

Thus every monomial α^E that can appear in the expansion has exponent

$$E = \sum_{k=0}^{s-1} u_k 2^k \quad \text{with} \quad 0 \le u_k \le t\cdot \mathrm{wt}_k(e).$$

Lemma 4.6 (Reachable exponent set)**.** *Let* $\mathsf{Reach}(e) \subseteq \mathbb{Z}_{\ge 0}$ *be the set containing any integer* E *for which there exists a choice of curve coefficients* $\{c_{i,j}\}$ *such that* α^E *appears with nonzero coefficient in* $M_e(\psi(\alpha))$*. Then*

$$\mathsf{Reach}(e) = \Big\{ \sum_{k=0}^{s-1} u_k 2^k : \ 0 \le u_k \le t\cdot \mathrm{wt}_k(e) \Big\}.$$

Proof. The inclusion $\mathsf{Reach}(e) \subseteq \{\sum u_k 2^k\}$ is the degree accounting above. For the converse, treat the coefficients $c_{i,j}$ as algebraically independent indeterminates. Fix any tuple $(u_0,\ldots,u_{s-1})$ in range; choose, for each bit k, a multiset of $\mathrm{wt}_k(e)$ integers in $\{0,\ldots,t\}$ summing to u_k (possible by the range condition). Then in the full expansion of $\prod_i \psi_i(\alpha)^{e_i}$, there is a specific product term obtained by selecting those chosen degrees at each factor, yielding a monomial in the indeterminates times $\alpha^{\sum_k u_k 2^k}$ with coefficient 1 (in characteristic two).

Notice that when α^E is produced by some choice of u_k, it cannot be "canceled" by a different choice and therefore must appear in the sum: $M_e(\psi(\alpha))$ is a product that runs over i,k and, in each product term, we sum over j. The formal expansion of $M_e(\psi(\alpha))$ is then a sum of products: every product is characterized by a unique choice of j-values $j(i,k)$ and is of the form $\prod_{i,k} c_{i,j(i,k)}^{2^k}$. Since we treat $c_{i,j}$ as independent indeterminates, the mapping between (i,j,k) and $c_{i,j}^{2^k}$ is 1-1. Hence, distinct mappings $(i,k) \mapsto j(i,k)$ give distinct monomials $\prod_{i,k} c_{i,j(i,k)}^{2^k}$ and no two monomials in

the sum are equivalent and there are no cancellations (since $2^k < q$, cancellations only come from equivalent degrees).

To conclude, the coefficient of α^E is a nonzero polynomial in the indeterminates, so it is nonzero for some assignment in $\mathbb{F}_q$. □

The reachability criteria from Lemma 4.6 ignores degree reductions modulu $\alpha^q - \alpha$, which we must take into account for deciding the degree of $M(\psi(\alpha))$ as a polynomial in the ring $\mathbb{F}_q[\alpha]/\langle\alpha^q - \alpha\rangle$, which actually determines membership in the Reed-Solomon code. To bridge upon the gap, we use the following characterization.

Lemma 4.7 (Reduction modulo $\alpha^q - \alpha$)**.** *For any integer $E \geq 1$,*

$$\alpha^E \equiv \alpha^{\,1+((E-1) \bmod (q-1))} \pmod{\alpha^q - \alpha}.$$

In particular, α^E reduces to a α^{q-1}-term iff $E > 0$ and $E \equiv 0 \pmod{q-1}$.

Proof. On $\mathbb{F}_q^*$ we have $\alpha^{q-1} = 1$, so $\alpha^E = \alpha^{1+(E-1)} \equiv \alpha^{1+((E-1) \bmod (q-1))}$. The congruence also holds in the quotient ring $\mathbb{F}_q[\alpha]/(\alpha^q - \alpha)$, and the stated criterion for α^{q-1} follows by setting $1 + ((E-1) \bmod (q-1)) = q-1$. □

By Lemma 4.7, a term α^E in $M_e(\psi(\alpha))$ reduces to a α^{q-1}-term iff $E > 0$ and $E \equiv 0 \pmod{q-1}$. Thus e is good iff for every curve ψ of degree at most t, the reduced polynomial has no α^{q-1} term. By Lemma 4.6, it suffices to check whether *any reachable exponent* hits the bad congruence class.

Lemma 4.8 (Goodness via Reachability)**.** *A monomial exponent vector $e \in \{0, \ldots, q-1\}^m$ is* bad *iff there exists $E \in \mathsf{Reach}(e)$ such that $E > 0$ and $E \equiv 0 \pmod{q-1}$. Equivalently, e is good iff the congruence class $0 \in \mathbb{Z}_{q-1}$ is* not *reachable by $\sum_k u_k 2^k$ with the above ranges, except via the trivial choice $u_k = 0$ for all k.*

Proof. If such an E exists, Lemma 4.6 gives a curve ψ where α^E appears with nonzero coefficient. By Lemma 4.7, this produces a nonzero α^{q-1}-coefficient in the reduced representative, so the restriction is not in $\mathsf{RS}(q, q-1)$ and e is bad. Conversely, if no such E exists, then no monomial term in the expansion can reduce to α^{q-1}, so every restriction lies in $\mathsf{RS}(q, q-1)$ and e is good. □

Having reduced the problem of testing good monomials to testing a reachability criteria, we implement in Algorithm 1 a dynamic program that takes as input a monomial and determines whether it is good or bad.

Algorithm 1 IsGoodMonomial$(e; q = 2^s, m, t)$

Input: $e \in \{0, \ldots, q-1\}^m$
Output: `good` or `bad`

Compute bit weights $\mathrm{wt}_k(e)$ for $k = 0, \ldots, s-1$;
Initialize a boolean array $S[0 \ldots q-2] \leftarrow 0$ and set $S[0] \leftarrow 1$;
For $k = 0, 1, \ldots, s-1$. Let $U_k := t \cdot \mathrm{wt}_k(e)$
. Initialize $S'[0 \ldots q-2] \leftarrow 0$
. **For** $r \in \{0, \ldots, q-2\}$ with $S[r] = 1$
. . **For** $u = 0, 1, \ldots, U_k$
. . . $S'[(r + u \cdot 2^k) \bmod q-1] \leftarrow 1$
. Set $S \leftarrow S'$
If $S[0] = 1$ and $\exists k$ with some $u_k > 0$ used along the DP path
. **Return** `bad`

Return `good`

To test a single monomial, the dynamic program maintains a length-$(q-1)$ boolean array through $s = \log_2 q$ stages. At stage k we loop over at most $q-1$ residues and over $U_k \leq tm$ choices, resulting in running time of $\tilde{O}(q \cdot tm)$ and space complexity of $O(q)$. To enumerate all good monomials, one could naively run Algorithm 1 over all monomials $e \in \{0, \ldots, q-1\}^m$, taking overall time $\tilde{O}(q^{m+1} \cdot tm)$. There is however a faster way for enumeration: observe that the goodness of a monomial e depends only on its weight vector $(\mathrm{wt}_1(e), \ldots, \mathrm{wt}_{s-1}(e))$. It is sufficient, then to run Algorithm 1 over all $(m+1)^s$ possible weight vectors and then select the good monomias as those corresponding to good weight vectors. This takes time $\tilde{O}(q^m + m^{s+1} \cdot q \cdot t)$.

4.2.2 Asymptotic Lower Bound on Rate

Since monomials are linearly independent as functions over $\mathbb{F}_q^m$, the dimension of the code $\mathsf{Lift\text{-}RS}_q(m, t)$ is equal to the number of good monomials (cf. Corollary 4.4) and its rate is

$$\frac{\dim_{\mathbb{F}_q}(\mathsf{Lift\text{-}RS}_q(m,t))}{q^m} = \frac{\#\{\text{good monomials}\}}{q^m}. \tag{2}$$

We prove the lower bound from Theorem 4.2 by upper-bounding the fraction of *bad* monomials. Our proof closely follows existing rate analyses for lifted RS codes from the literature [GKS13, HPPV20, LN21].

Define $B := tm$ and assume $q > B$ (equivalently $s > \log_2 B$; recall $q = 2^s$). Let

$$L := \lfloor \log_2(B+1) \rfloor + 1 \tag{3}$$

and define

$$\varepsilon(m,t) := m - \frac{1}{L}\log_2\big(2^{mL} - 1\big) = -\frac{1}{L}\log_2\big(1 - 2^{-mL}\big) > 0. \tag{4}$$

First, let us confirm that $\varepsilon(m,t)$ admits the explicit lower bound from the statement of Theorem 4.2:

$$\varepsilon(m,t) \geq \frac{2^{-tm^2}}{tm \ln 2}. \tag{5}$$

Indeed, by the inequality $-\ln(1-x) \geq x$ for $x \in (0,1)$, we have

$$\varepsilon(m,t) = -\frac{1}{L}\log_2(1 - 2^{-mL}) = \frac{1}{L \ln 2}\big(-\ln(1 - 2^{-mL})\big) \geq \frac{2^{-mL}}{L \ln 2}.$$

Since $m \geq 2$ and $t \geq 1$, we have $B = tm \geq 2$ and hence $L \leq B$ (because $2^B > B+1$ for $B \geq 2$, so $\log_2(B+1) < B$ and thus $\lfloor \log_2(B+1) \rfloor + 1 \leq B$). Therefore $\frac{1}{L} \geq \frac{1}{B} = \frac{1}{tm}$ and also $mL \leq mB = tm^2$, which implies $2^{-mL} \geq 2^{-tm^2}$. Combining yields (5).

We proceed to bound the rate in three steps.

Step 1: from badness to a multiple of $q-1$. Fix an exponent vector $e \in \{0, \ldots, q-1\}^m$ and recall the notation $M_e(\alpha_1, \ldots, \alpha_m) = \prod_{i=1}^m \alpha_i^{e_i}$. By Lemma 4.8, e is bad iff there exist integers $u_0, \ldots, u_{s-1}$ with $0 \leq u_k \leq t \operatorname{wt}_k(e)$ such that

$$E := \sum_{k=0}^{s-1} u_k 2^k \text{ satisfies } E > 0 \text{ and } E \equiv 0 \pmod{q-1}. \tag{6}$$

Since $\operatorname{wt}_k(e) \leq m$, we have $u_k \leq tm = B$ for all k, and thus $E \leq B \sum_{k=0}^{s-1} 2^k = B(q-1)$. Therefore the congruence in (6) forces

$$E = a(q-1) \quad \text{for some integer } a \in \{1, 2, \ldots, B\}. \tag{7}$$

We call e *a-bad* if (7) holds for that particular a. Then $\#\{\text{bad monomials}\} \leq \sum_{a=1}^{B} \#\{a\text{-bad monomials}\}$ by a union bound.

Step 2: a run of L zero bit-columns is impossible for any a. Fix $a \in \{1, \ldots, B\}$ and let $r := \lceil \log_2 a \rceil$. Because $q > B \geq a$, we have $r \leq s$. Write the target integer $T := a(q-1) = a(2^s - 1) = a2^s - a$. Modulo 2^s, this equals $2^s - a = (2^s - 1) - (a-1)$. Hence the binary expansion of $2^s - a$ has the property that its bits are all 1 in positions $k \geq r$ (since $a-1$ has no nonzero bits at positions $\geq r$). Let $b_k \in \{0,1\}$ denote the k-th bit of $2^s - a$, so $b_k = 1$ for all $k \geq r$.

Assume e is a-bad, witnessed by some choice of u_k giving $E = T$. Consider the standard carry recursion for expressing the equality $\sum_{k=0}^{s-1} u_k 2^k = T$ in base 2: there exist integers ("carries") $c_0, \ldots, c_s$ with $c_0 = 0$ such that for every $k = 0, \ldots, s-1$,

$$u_k + c_k = b_k + 2c_{k+1}. \tag{8}$$

Carry bound. We claim that $0 \leq c_k \leq B$ for all k. Indeed, $c_0 = 0$ and if $0 \leq c_k \leq B$ then using $u_k \leq B$ and $b_k \geq 0$ in (8),

$$2c_{k+1} = u_k + c_k - b_k \leq u_k + c_k \leq B + B = 2B,$$

so $c_{k+1} \leq B$; also $u_k + c_k \geq b_k$ implies $c_{k+1} \geq 0$. Thus by induction $0 \leq c_k \leq B$ for all k.

Run-of-zeros contradiction. If at some position $k \geq r$ we have $\operatorname{wt}_k(e) = 0$, then $u_k \leq t\operatorname{wt}_k(e) = 0$, hence $u_k = 0$. Since also $b_k = 1$ for $k \geq r$, the carry equation (8) becomes $c_k = 1 + 2c_{k+1}$, which forces c_k to be odd and yields $c_k \geq 1$. If this happens for ℓ consecutive positions $k, k+1, \ldots, k+\ell-1$ (all with $k \geq r$ and all having $\operatorname{wt}.(e) = 0$), then iterating $c_i = 1 + 2c_{i+1}$ gives

$$c_k = (2^\ell - 1) + 2^\ell c_{k+\ell} \;\geq\; 2^\ell - 1.$$

By the carry bound $c_k \leq B$; therefore such a run is possible only if $2^\ell - 1 \leq B$. With L as in (3), we have $2^L > B + 1$ and hence $2^L - 1 > B$. Thus in the suffix positions $\{r, r+1, \ldots, s-1\}$ there cannot be a run of L consecutive indices with $\operatorname{wt}_k(e) = 0$. Equivalently, in the bit-column sequence $v_k = (e_{1,k}, \ldots, e_{m,k}) \in \{0,1\}^m$, there is no run of L consecutive all-zero columns in that suffix.

Step 3: count bit-column sequences avoiding L consecutive zero columns. Let $A := 2^m$ be the number of possible bit-columns $v_k \in \{0,1\}^m$. For $n \geq 0$, let $N(n)$ denote the number of length-n words over an alphabet of size A that avoid the forbidden block of L consecutive all-zero columns. We claim the crude bound

$$N(n) \;\leq\; A^L\,(A^L-1)^{\lceil n/L\rceil}. \tag{9}$$

To see this, write $n = qL + \rho$ with $0 \leq \rho < L$ and split the word into a prefix of length ρ followed by q aligned blocks of length L. If the word avoids L consecutive zeros, then in particular none of these aligned blocks can be all-zero, so each block has at most $A^L - 1$ choices; the prefix has at most $A^\rho \leq A^L$ choices. This proves (9).

Now fix a and let $r = \lceil \log_2 a \rceil$ and $n := s - r$. By Step 2, any a-bad monomial e must have its suffix column sequence $(v_r, \ldots, v_{s-1})$ belonging to this avoidance set. Therefore

$$\#\{a\text{-bad } e\} \;\leq\; A^r\,N(n).$$

Since the total number of reduced monomials is $A^s = q^m$, the fraction of a-bad monomials is at most $A^r N(n)/A^s = N(n)/A^n$. Using (9) and the inequality $\lceil n/L\rceil \leq n/L + 1$ gives

$$\frac{N(n)}{A^n} \;\leq\; \frac{A^L\,(A^L-1)^{n/L+1}}{A^n} = \frac{A^L\,(A^L-1)}{A^n}\cdot\left(\frac{(A^L-1)^{1/L}}{A}\right)^n \;\leq\; A^{2L}\left(\frac{(A^L-1)^{1/L}}{A}\right)^n,$$

where we used $A^L - 1 \leq A^L$. Let $\gamma := (A^L-1)^{1/L}$. Then $\gamma < A$ and the above becomes $A^{2L}(\gamma/A)^n$. Since $n = s - r \geq s - \lceil \log_2 B\rceil$ and $(\gamma/A) < 1$, we further have

$$\left(\frac{\gamma}{A}\right)^n \;\leq\; \left(\frac{\gamma}{A}\right)^{s-\lceil\log_2 B\rceil} = \left(\frac{\gamma}{A}\right)^s\cdot\left(\frac{\gamma}{A}\right)^{-\lceil\log_2 B\rceil} \;\leq\; \left(\frac{\gamma}{A}\right)^s\cdot A^{\lceil\log_2 B\rceil} \;\leq\; \left(\frac{\gamma}{A}\right)^s\cdot(2B)^m,$$

where we used $(\gamma/A)^{-1} = A/\gamma \leq A$ and $A^{\lceil\log_2 B\rceil} \leq A^{\log_2(2B)} = (2B)^m$.

Finally, observe that

$$\left(\frac{\gamma}{A}\right)^s = 2^{s(\log_2\gamma - \log_2 A)} = 2^{-s\left(m - \frac{1}{L}\log_2(A^L-1)\right)} = q^{-\varepsilon(m,t)}$$

with $\varepsilon(m,t)$ defined in (4). Putting everything together, for each a the a-bad fraction is at most $2^{2mL}(2B)^m q^{-\varepsilon(m,t)}$, since $A = 2^m$. Summing over $a \in \{1,\ldots,B\}$ gives

$$1 - \mathrm{rate}(C_{m,q,t}) \;\leq\; \frac{\#\{\text{bad monomials}\}}{q^m} \;\leq\; B\cdot 2^{2mL}(2B)^m\cdot q^{-\varepsilon(m,t)} = tm\cdot 2^{2mL}(2tm)^m\cdot q^{-\varepsilon(m,t)}.$$

Using $L \leq B = tm$ we have $2^{2mL} \leq 2^{2tm^2}$. Also $(2tm)^m = 2^{m\log_2(2tm)} \leq 2^{m\cdot 2tm} = 2^{2tm^2}$ since $\log_2 x \leq x$ for $x \geq 1$. Thus $2^{2mL}(2tm)^m \leq 2^{4tm^2}$, giving the bound stated in Theorem 4.2 and completing the proof of the theorem.

4.2.3 Concrete Rate Estimation

A practical way to estimate the actual concrete rate of $\mathsf{Lift\text{-}RS}_q(m,t)$ for selected regimes of parameters a is Monte-Carlo experiment: sample random exponent vectors $e \in \{0,\ldots,q-1\}^m$, test goodness using Algorithm 1, and take the fraction declared `good`.

We present the experimental results in Table 3 and contrast the estimated rate with the rate of RM codes, which approaches $1/(m!\cdot t^m)$ (Corollary 4.1).

4.2.4 Encoding in Quasi-Linear Time

Lemma 4.9 (Lifted-RS Encoding)**.** $\mathsf{Lift\text{-}RS}_q(m, \ell, t)$ *has a systematic encoding function that is computable in time* $\tilde{O}(m \cdot q^m)$*.*

The encoding algorithm assumes a pre-processing step that runs in time $\tilde{O}(q^m + m^{s+1} \cdot q \cdot t)$ *and succeeds except with probability negligible in* q^m*.*

A starting point towards a systematic encoder is to identify an explicit *interpolating set*, namely a set of coordinates that can constitute the systematic part of the codeword, i.e. the part that contains the message.

Definition 4.5 (Interpolating Set)**.** *A set* $S \subseteq \mathbb{F}_q^m$ *is an* interpolating set *for a code* C *if* $\dim(C|_S) = \dim(C)$*. Equivalently, prescribing the values of a codeword on* S *gives a unique codeword.*

Guo and Kopparty [GK14, Theorem A.1] identify a universal interpolating set for affine-invariant codes through an isomorphism to the extension field where the code becomes univariate.

Fix an $\mathbb{F}_q$-linear isomorphism

$$\varphi : \mathbb{F}_{q^m} \mapsto \mathbb{F}_q^m.$$

A convenient explicit choice uses the trace map $\mathrm{Tr} : \mathbb{F}_{q^m} \to \mathbb{F}_q$, where $\mathrm{Tr}(z) = \sum_{i=0}^{m-1} z^{q^i}$ (see [GK14, Section 2.5]): Pick $\alpha_1, \ldots, \alpha_m \in \mathbb{F}_{q^m}$ linearly independent over $\mathbb{F}_q$ and set

$$\varphi(z) = (\mathrm{Tr}(\alpha_1 z), \ldots, \mathrm{Tr}(\alpha_m z)).$$

Note that φ and its inverse φ^{-1} are computable in time $\tilde{O}(m)$.

Theorem 4.3 ([GK14])**.** *Let* $C \subseteq \{\mathbb{F}_q^m \to \mathbb{F}_q\}$ *be a nontrivial affine-invariant code of dimension* k*. Let* $\omega \in \mathbb{F}_{q^m}^*$ *be of order* $q^m - 1$*, i.e. a generator. Let* $S = \{\omega, \omega^2, \ldots, \omega^k\} \subseteq \mathbb{F}_{q^m}$*. Then,* $\varphi(S) \subseteq \mathbb{F}_q^m$ *is an interpolating set for* C*.*

Given a message $\mathbf{x} \in \mathbb{F}_q^k$, there exists then a unique $\mathbf{c} \in \mathsf{Lift\text{-}RS}_q(m, \ell, t)$ such that

$$\mathbf{c}(\varphi(\omega^j)) = \mathbf{x}_j \quad \text{for } j = 1, \ldots, k.$$

To obtain a systematic encoding algorithm, two steps remain: First, to identify the codeword $\mathbf{c}$ that systematically encodes $\mathbf{x}$ as above (through, say, its basis coefficients). Second, to compute all values in $\mathbf{c}$.

We show how to perform both of these tasks in quasi-linear time over the univariate code obtained by applying φ to the coordinate space of $\mathsf{Lift\text{-}RS}_q(m, \ell, t)$. Since φ is efficiently computable, we obtain a quasi-linear-time systematic encoder for $\mathsf{Lift\text{-}RS}_q(m, \ell, t)$ as well.

Explicitly, we look at the code obtained by applying φ over the coordinate space of lifted RS codewords:

$$C' = \{\mathbf{c} \circ \varphi \mid \mathbf{c} \in \mathsf{Lift\text{-}RS}_q(m, \ell, t)\} \subset \{\mathbb{F}_{q^m} \to \mathbb{F}_q\}.$$

By Claim 4.1, C' is affine-invariant over the 1-dimensional $\mathbb{F}_q$-vector space $\mathbb{F}_{q^m}$. By Lemma 4.5, then, there exists a basis of univariate monomials that spans C'. Therefore, we may write

$$C' = \mathbf{Span}_{\mathbb{F}_q}\big(\{f_i(z) = z^i \mid i \in D'\} \big)$$

for some *degree set* $D' \subset \{0, \ldots, Q-1\}$ of size $k = \dim(\mathsf{Lift\text{-}RS}_q(m, \ell, t))$.

To use C' algorithmically we need to compute the degree set D'. First, we enumerate the basis of good monomials in time $\tilde{O}(q^m + m^{s+1} \cdot q \cdot t)$ by running Algorithm 1 over all possible weight vectors as described above (Section 4.2.1). This allows us to sample random codewords from the lifted code and, therefore, from C'. Given this, a practical method to find D' is to take unions of supports of random codewords in C'.

Claim 4.2 (Probabilistically Computing D')**.** *There exists a probabilistic algorithm that runs in time $\tilde{O}(T \cdot m \cdot q^m)$ and outputs D' with probability at least $1 - q^{m-T}$*

Proof. It is sufficient to show that for any fixed $i \in D'$, a uniformly random $f \leftarrow C'$ satisfies $\Pr[i \notin \mathrm{Supp}(f)] \leq 1/q$. Consequently, drawing T independent random codewords and taking the union of their supports misses i with probability at most q^{-T}. By union bound, T samples miss *some* $i \in D'$ with probability at most $|D'| \cdot q^{-T} < q^{m-T}$.

The above can be done in runtime $\tilde{O}(T \cdot m \cdot q^m)$: Sampling f is simply done by sampling a polynomial $\mathbf{c} \in \mathsf{Lift\text{-}RS}_q(m, \ell, t)$ by sampling good monomial coefficients, then evaluating $\mathbf{c}$ on $\mathbb{F}_q^m$ then using the isomorphism φ to compute the evaluation of f over $\mathbb{F}_{q^m}$. Given the evaluations of f, its monomials are computed by polynomial interpolation. Both multivariate polynomial evaluation over all points and univariate polynomial interpolation can be computed in quasi-linear time $\tilde{O}(m \cdot q^m)$ using FFT techniques (and, for the former, by exploiting the tensor structure of multivariate polynomials); see, e.g. [MB72].

To bound the probability of a miss, let us fix $i \in D'$. There exists some codeword $f^\star \in C'$ whose coefficient $f_i^\star \neq 0$. The map $f \mapsto f_i$ is $\mathbb{F}_q$-linear and not identically zero. Thus its kernel is a subspace of co-dimension at least 1 over $\mathbb{F}_q$, so a uniform $f \in C'$ lands in the kernel with probability at most $1/q$. □

We are now prepared to describe the systematic encoding algorithm. At a high level, the encoding uses the univariate perspective captured by the code C', where we know a well-structured interpolation set (Theorem 4.3) that allows to use FFT-based interpolation and evaluation algorithms. We assume that we have already computed the dimension $k = \dim(\mathsf{Lift\text{-}RS}_q(m, \ell, t))$ and the degree set D' of the code C'. On input $\mathbf{x} \in \mathbb{F}_q^k$, encoding proceeds as follows:

1. Find $a_1, \ldots, a_k \in \mathbb{F}_{q^m}$ such that $f(z) = \sum_{i \in D'} a_i z^i$ satisfies $f(\omega^j) = \mathbf{x}_j$ for $j = 1, \ldots, k$. This corresponds to solving a linear-equation system where the matrix is a generalized Vandermonde matrix of the form $((\omega^j)^{i_\ell})_{\substack{1 \leq j \leq k \\ 1 \leq \ell \leq k}}$, where $\{i_1, \ldots, i_k\} = D'$ are distinct. This can be done in time quasi-linear in k (see, e.g. [BS04, BJS07]).

2. Compute $f(z)$ for all $z \in \mathbb{F}_{q^m}$ in time $\tilde{O}(q^m)$ using quasi-linear FFT-based evaluation (as in Claim 4.2, see [MB72]).

3. Output $\mathbf{c}$ defined by $\mathbf{c}(z) = f(\varphi^{-1}(z))$. This can be done in time $\tilde{O}(m \cdot q^m)$.

5 Doubly-Efficient sk-PIR with Low Storage

We build doubly-efficient sk-PIR from smooth locally decodable codes, by generalizing the paradigm from [BIPW17, CHR17] and instantiating it using the LDC constructions from Section 4: $\mathsf{LDC}_{\mathsf{RM\text{-}Conc}}$ (Corollary 4.2) and $\mathsf{LDC}_{\mathsf{Lift\text{-}RS}}$ (Corollary 4.3).

5.1 Secret-key PIR from Smooth Locally Decodable Codes

We recall the general approach for building doubly-efficient sk-PIR from families of locally decodable codes [BIPW17, CHR17]. At pre-processing, the database is encoded via a random locally decodable code from the family $\mathsf{LDC} \leftarrow \mathcal{LDC}$, which is sampled using the client's secret key (Definition 3.4). To retrieve a database element, the client generates a query set Q using the local decoder of LDC. The client then sends to the server a random larger set Q', of size $L \geq |Q|$, such that $Q \subset Q'$. This is a modification compared to the original works: in [BIPW17] the client sends Q directly, while

in [CHR17] the additional coordinates $Q' \setminus Q$ have fixed positions in the client's queries (which are sampled by the client and are hidden from the server). The server, in turn, replies with the corresponding database values, allowing the client to compute the desired value using the decoding algorithm of the LDC.

We describe the construction formally in Fig. 5. We note a minor gap in syntax relative to our definition of sk-PIR (Definition 3.2): the input to the construction is a database of size k over the finite-field alphabet $\mathbb{F}$ defined by the underlying LDC family. Any sk-PIR over a large alphabet can be easily transformed to sk-PIR over bits, on par with Definition 3.2: Write the database as an array of field elements by grouping every $\lceil \log |\mathbb{F}| \rceil$ bits. To retrieve a database bit, the client retrieves the field element that contains it.

Parameters:

- A family of (k, n, ℓ)-LDC $\mathcal{LDC} = (\mathsf{G}, \mathsf{E}, \mathsf{Q}, \mathsf{D})$ over a finite field $\mathbb{F} := \mathbb{F}(\lambda)$.
- A query size parameter $L(\lambda) \geq \ell(\lambda)$.
- A pseudorandom function $\mathsf{PRF} : \{0,1\}^\lambda \times \{0,1\}^\lambda \to \mathbb{F}$.

The Protocol:

- $\mathsf{Gen}(1^\lambda, 1^k)$: Output a uniform PRF key $\mathsf{sk} \leftarrow \{0,1\}^\lambda$.
- $\mathsf{Enc}(\mathsf{sk}, x \in \mathbb{F}^k)$: Choose the smallest $\lambda' \geq \lambda$ such that $k \leq k(\lambda')$. If $k < k(\lambda')$, make x of size $k(\lambda')$ by padding it with zeros.
  1. Use the outputs of $\mathsf{PRF}(\mathsf{sk}, (0, \cdot))$ to sample $\mathsf{id} \leftarrow \mathcal{LDC}.\mathsf{G}(1^{\lambda'})$ and denote $\mathsf{LDC} := \mathsf{LDC}_{\mathsf{id}}$.
  2. Output the following database encoding
$$\widehat{x} = \mathsf{LDC.E}(x) + \left(\mathsf{PRF}(\mathsf{sk}, (1, i)) \right)_{i=1,\ldots,n}.$$
- $\mathsf{Query}(\mathsf{sk}, i)$: Sample $(Q, \mathsf{aux}) \leftarrow \mathsf{LDC.Q}(i)$ and a uniformly random subset Q' of size $L(\lambda')$ conditioned on $Q \subseteq Q'$, and output
$$(\mathsf{que} = Q', \ \ \mathsf{aux}).$$
- $\mathsf{Ans}(\widehat{x}, \mathsf{que} = Q')$: Return $\widehat{x}_{Q'}$.
- $\mathsf{Dec}(\mathsf{sk}, i, \mathsf{ans} = \widehat{x}_{Q'}, \mathsf{aux})$: Output $\mathsf{LDC.D}(i, \widehat{x}_Q - \mathsf{PRF}(\mathsf{sk}, (1, j))_{j \in Q}, \mathsf{aux})$.

Figure 5: Doubly Efficient sk-PIR from Locally Decodable Codes.

Security of the sk-PIR reduces directly to the following distinguishing game between an adversary and a challenger (up to the security of the PRF):

1. The challenger samples a random locally decodable code from the family $\mathsf{LDC} \leftarrow \mathcal{LDC}$.
2. Given input $i \in [k]$ chosen by the adversary, the challenger samples a query set $(Q, \cdot) \leftarrow \mathsf{LDC.Q}(i)$ of size ℓ and hides it inside a uniformly random subset Q' of size $|Q'| = L \geq \ell$ such that $Q \subseteq Q'$. The challenger sends Q' back to the adversary.
3. The adversary wins if it can distinguish between the case where it is given samples $Q'_1, Q'_2, \ldots$ corresponding to its chosen inputs $i_1, i_2, \ldots$ as above and the case where it is given samples corresponding to the constant sequence $1, 1, \ldots$.

In the following, we define a notion of *sk-PIR friendliness* for families of locally decodable codes, that captures the case when the above distinguishing game is hard.

Definition 5.1 (sk-PIR Friendly LDC)**.** *We say that a (k, n, ℓ)-LDC family $\mathcal{LDC}$ is* sk-PIR friendly *with a polynomially bounded parameter $L \geq \ell$ if the sk-PIR construction from Fig. 5, when instantiated with parameters $\mathcal{LDC}$, L and any pseudorandom function* PRF*, is secure.*

Given an sk-PIR friendly family of (k, n, ℓ)-LDCs with parameter L, we obtain a sk-PIR scheme with communication complexity and server-side complexity $O(L)$, and storage rate k/n. When $L = o(k)$, the scheme is doubly-efficient.

Generally speaking, to obtain hardness from a family of LDCs, it is necessary that the distribution $\mathsf{LDC} \leftarrow \mathcal{LDC}$ has a sufficient amount of entropy. One generic way to produce entropy is to take an explicit LDC and randomize it by applying a uniformly random permutation over its coordinates. This approach was taken in [BIPW17, CHR17], where the sk-PIR construction is based on permuted RM codes, and we follow it in this work to obtain new families of LDCs that we conjecture to be sk-PIR friendly.

Definition 5.2 (Code Permutation)**.** *Let $\mathsf{LDC} = (\mathsf{E}, \mathsf{Q}, \mathsf{D})$ be an explicit (k, n, ℓ)-locally decodable code. We define the LDC family of* permuted LDC*, which we denote by* pLDC*, to be the LDC family where:*

- $\mathsf{pLDC.G}(1^\lambda)$ *outputs a uniformly random permutation $\pi : [n] \to [n]$.*
- $\mathsf{pLDC.E}(\pi, \mathbf{x})$ *computes $\mathbf{c} = \mathsf{LDC.E}(\mathbf{x})$ and outputs $\mathbf{c}' = (\mathbf{c}_{\pi(1)}, \ldots, \mathbf{c}_{\pi(n)})$.*
- $\mathsf{pLDC.Q}(\pi, i)$ *samples $(Q = (q_1, \ldots, q_\ell), \mathsf{aux}) \leftarrow \mathsf{LDC}(\mathsf{Q}(i))$ and outputs $\pi(Q) = (\pi(q_1), \ldots, \pi(q_\ell))$ and* aux*.*
- $\mathsf{pLDC.D}(\pi, i, \mathbf{c}'_{\pi(Q)} = \mathbf{c}_Q, \mathsf{aux})$ *outputs* $\mathsf{LDC.D}(i, \mathbf{c}_Q, \mathsf{aux})$*.*

Another design principle that characterizes the approach of [BIPW17, CHR17] is the focus on smooth LDCs (Definition 4.1). As observed in these works, in an LDC that is not smooth, the distribution of t query points may depend on the identity of the decoded element. This makes the sk-PIR susceptible to statistical attacks that collect t-local statistics and exploit the disparity in the pdf to break security. The complexity of such an attack grows with q^t and, therefore, we will require any potential sk-PIR friendly LDC to have a super-constant smoothness parameter t. While [BIPW17, CHR17] consider only the strong notion of smoothness, we note that the weak notion of smoothness is already sufficient to defy the aforementioned attack as we explain in Section 6.2.6.

5.2 Instantiation 1: Lifted RS Codes

In our first instantiation, we use permuted lifted RS codes, namely the permuted variant of the the $\mathsf{Lift\text{-}RS}_q(m, t)$ LDC – $\mathsf{pLDC}_{\mathsf{Lift\text{-}RS}}$ (cf. Corollary 4.3 and Definition 5.2).

Recall that the local decoder of $\mathsf{LDC}_{\mathsf{Lift\text{-}RS}}$ is identical to the curve-based local decoder of RM codes (Fig. 3). Hence, the sk-PIR friendliness of permuted $\mathsf{LDC}_{\mathsf{Lift\text{-}RS}}$ is equivalent to the sk-PIR friendliness of permuted $\mathsf{LDC}_{\mathsf{RM\text{-}RS}}$, since the notion depends solely on the local decoder's query distribution.

Permuted $\mathsf{LDC}_{\mathsf{RM\text{-}RS}}$ codes were used in the original sk-PIR construction [BIPW17, CHR17]. In [BIPW17], the sk-PIR friendliness of these codes with parameter $L = \ell$ is abstracted as a *permuted low-degree puzzle*: the problem of distinguishing lists of points coming from random permuted low-degree curves against lists that are sampled uniformly at random.

Definition 5.3 (Permuted Low-Degree Puzzle [BIPW17])**.** *Let $m := m(\lambda)$, $t := t(\lambda)$ and $\ell := \ell(\lambda)$ be polynomially bounded and let $q := q(\lambda)$ be a prime power such that $\log q = \mathrm{poly}(\lambda)$.*

The permuted low-degree puzzle (m, t, ℓ)-PLD_q *is the problem of distinguishing between two oracles: $\mathsf{PLD} = \{\mathsf{PLD}_\pi\}$, parameterized by a permutation $\pi : \mathbb{F}_q^m \to \mathbb{F}_q^m$, and $\mathsf{Rand} := \mathsf{Rand}(\lambda)$.*

- $\mathsf{PLD}_\pi(i \in \mathbb{F}_q^m)$*:*
  1. *Sample a uniformly random degree-t parametric curve $\psi : \mathbb{F}_q \to \mathbb{F}_q^m$ such that $\psi(0) = i$.*
  2. *Sample uniform and distinct $z_1, \ldots, z_\ell \leftarrow \mathbb{F}_q^*$.*
  3. *Output $\pi(\psi(z_1)), \ldots, \pi(\psi(z_\ell))$.*
- $\mathsf{Rand}(i \in \mathbb{F}_q^m)$*: output uniformly random $y_1, \ldots, y_\ell \leftarrow \mathbb{F}_q^m$.*

We say that a PLD *instance is* (T, ε)-hard *if for any non-uniform oracle-aided algorithm $\mathcal{A}(1^\lambda)$ that runs in time $T(\lambda)$, it holds*

$$|\Pr[\mathcal{A}^{\mathsf{PLD}_\pi}(1^\lambda) = 1] - \Pr[\mathcal{A}^{\mathsf{Rand}}(1^\lambda) = 1]| \leq \varepsilon(\lambda),$$

for a uniformly random permutation $\pi : \mathbb{F}_q^m \to \mathbb{F}_q^m$.

We say that PLD *is* hard *if it is (T, ε)-hard for any $T = \mathrm{poly}(\lambda)$ and $\varepsilon = 1/\mathrm{poly}(\lambda)$.*

The permuted low-degree puzzle is a special case of a generalized framework of *permuted puzzle problems* that was subsequently developed in [BHW19].

The sk-PIR friendliness of t-smooth permuted $\mathsf{LDC}_{\mathsf{RM\text{-}RS}}$ with $L = \ell$ is immediately implied by the polynomial hardness of $(m, t, \ell = dt + 1)$-PLD (where m, d are the RM parameters). The only reason why equivalence does not hold is that, in PLD, the curve points are given as a list while in the sk-PIR security game, the adversary only sees the set of distinct points.

[BIPW17] conjecture that no polynomial-time adversary is able to solve the permuted low-degree puzzle with advantage that exceeds a negligible function in t when $\ell < q - 1$.[5] We consider a more conservative variant of the conjecture, where we require the gap $q - \ell$ to grow with the security parameter, due to potential algebraic attacks that may exploit the low entropy in the subset of values $z_1, \ldots, z_\ell$ (see Section 6.1.2 for details).

Conjecture 5.1 (Asymptotic Hardness of PLD)**.** *For any polynomially bounded $m(\lambda), t(\lambda), q(\lambda), \ell(\lambda)$ such that $m \geq 2$, $t = \omega(1)$, $q = \lambda^{\Omega(1)}$ and $\ell < q - \lambda$, (m, t, ℓ)-PLD_q is hard.*

Instantiating the sk-PIR of Fig. 5 using the permuted RM code $\mathsf{LDC}_{\mathsf{RM\text{-}RS}}$ (Proposition 4.1) with a sufficiently large constant $m \geq 2$, any $t = \omega(1)$, sufficiently large $q = O(t \cdot k^{1/m})$ and $\ell < q - 1$, and $L = \ell$, gives a doubly-efficient secure sk-PIR under the PLD conjecture. The sk-PIR inherits a low storage rate, that vanishes with λ, from the RM code (Corollary 4.1).

Theorem 5.1 (Doubly-Efficient sk-PIR from PLD [BIPW17])**.** *Assuming PLD is hard (Conjecture 5.1), for any constant $\varepsilon > 0$, there exists a doubly-efficient sk-PIR scheme with communication and server work k^ε, and storage rate that vanishes with λ.*

[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 $\ell = q - 1$.

To obtain doubly-efficient sk-PIR with high storage rate, we replace the RM encoding with lifted RS encoding. The lifted RS code that give us hight rate are based on Reed-Solomon codes with dimension $q-1$ and consequently have locality $\ell = q-1$ (cf. Theorem 4.2 and Corollary 4.3).

The PLD problem becomes easy whenever $\ell = q-1$, as the choice of of points $\{z_1, \ldots, z_\ell\} = \mathbb{F}_q^*$ is fixed and no longer hidden, allowing for the aforementioned algebraic attack (Footnote 5 and Section 6.1.2). To obstruct the algebraic structure when $\ell = q-1$, we inject noise to the sk-PIR queries by setting $L > \ell$.

The sk-PIR friendliness of curve-based decoding with $L > \ell$ immediately reduces to a noisy variant of the PLD puzzle, where the ℓ points of the curve are mixed with additional uniformly random points in a list of length $L > \ell$. We call this variant *permuted low-degree with noise*, or PLDN for short.

Definition 5.4 (Permuted Low-Degree with Noise)**.** *Let $m := m(\lambda)$, $t := t(\lambda)$, $\ell := \ell(\lambda)$ and $L := L(\lambda) > \ell$ be polynomially bounded and let $q := q(\lambda)$ be a prime power such that $\log q = \mathrm{poly}(\lambda)$.*

The permuted low-degree with noise *problem (m, t, ℓ, L)-PLDN_q is the problem of distinguishing between two oracles: $\mathsf{PLDN} = \{\mathsf{PLDN}_\pi\}$, parameterized by a permutation $\pi : \mathbb{F}_q^m \to \mathbb{F}_q^m$, and $\mathsf{Rand} := \mathsf{Rand}(\lambda)$.*

- $\mathsf{PLDN}_\pi(i \in \mathbb{F}_q^m)$*:*
  1. *Sample a uniformly random degree-t parametric curve $\psi : \mathbb{F}_q \to \mathbb{F}_q^m$ such that $\psi(0) = i$.*
  2. *Sample uniform and distinct $z_1, \ldots, z_\ell \leftarrow \mathbb{F}_q^*$.*
  3. *Let $y_i = \pi(\psi(z_i))$ for $i = 1, \ldots, \ell$ and sample $y_{\ell+1}, \ldots, y_L \leftarrow \mathbb{F}_q^m$ uniformly at random.*
  4. *Sample a uniformly random permutation $f : [L] \to [L]$.*
  5. *Output $y_{f(1)}, \ldots, y_{f(L)}$.*
- $\mathsf{Rand}(i \in \mathbb{F}_q^m)$*: output uniformly random $y_1, \ldots, y_L \leftarrow \mathbb{F}_q^m$.*

*We say that a PLDN instance is (T, ε)-*hard *if for any non-uniform oracle-aided algorithm $\mathcal{A}(1^\lambda)$ that runs in time $T(\lambda)$, it holds*

$$|\Pr[\mathcal{A}^{\mathsf{PLDN}_\pi}(1^\lambda) = 1] - \Pr[\mathcal{A}^{\mathsf{Rand}}(1^\lambda) = 1]| \leq \varepsilon(\lambda),$$

for a uniformly random permutation $\pi : \mathbb{F}_q^m \to \mathbb{F}_q^m$.

We say that PLDN is hard *if it is (T, ε)-hard for any $T = \mathrm{poly}(\lambda)$ and $\varepsilon = 1/\mathrm{poly}(\lambda)$.*

Unlike PLD which completely breaks when $\ell = q-1$, we conjecture that the PLDN problem is polynomially hard, for sufficiently large choices of q and t, as long as L is sufficiently larger than ℓ. Once again, we adhere to a conservative regime and require $L - \ell > \lambda$.

Conjecture 5.2 (Asymptotic Hardness of PLDN)**.** *For any polynomially bounded $m(\lambda), t(\lambda), q(\lambda), \ell(\lambda)$ such that $m \geq 2$, $t = \omega(1)$, $q = \lambda^{\Omega(1)}$ and $\ell < L - \lambda$, (m, t, ℓ, L)-PLDN_q is hard.*

In Section 6, we support the PLD and PLDN conjectures by discussing how various known attack strategies, including algebraic and statistical/combinatorial attacks, fail in approaching the problems. We also draw connections to related cryptographic assumptions in the literature.

Conjecture 5.2 immediately implies the sk-PIR friendliness of $\mathsf{LDC}_{\mathsf{Lift\text{-}RS}}$ whenever $q = \lambda^{\Omega(1)}$ and any $t = \omega(1)$. Since Corollary 4.3 gives rate-1 with appropriate choice of super-constant t and m, e.g., $t = \Omega(\sqrt{\log\log\lambda})$ and $m = O((\log\log\lambda)^{1/5})$, we obtain a doubly-efficient sk-PIR scheme where the rate approaches 1 and server work is $k^{o(1)}$.

Theorem 5.2 (Doubly-Efficient sk-PIR from PLDN)**.** *Assuming PLDN is hard (Conjecture 5.2), there exists a doubly-efficient sk-PIR scheme with communication and server work $k^{o(1)}$, and storage rate approaching* 1*.*

The PLDN problem is closely related to the *hidden permutation with noise* (HPN) problem defined in [CHR17]. In HPN, the samples consist of points on a permuted random curve ψ, namely $\pi(\psi(z_1)), \ldots, \pi(\psi(z_L))$, for fixed $z_1, \ldots, z_L \in \mathbb{F}_q^*$, then the values at a fixed random subset of locations $S \subset [L]$ are replaced with uniformly random values. In other words, each sample is a mixture of $L - |S|$ curve points and $|S|$ random points, where the random points are always at the same positions in the sample. There are two differences compared to PLDN: First, the conjecture in [CHR17] is restricted to $\ell < L < q$ by the syntax of the sampler, while we do not impose this restriction and in fact violate it by $\ell = q - 1$ to obtain Theorem 5.2. Second, the noise is always injected at the same positions as S is fixed for all samples, while in PLDN we use a random permutation f to randomize the order. An HPN variant where we allow $\ell = q - 1$ implies the corresponding special case of Conjecture 5.2 (with $\ell = q - 1$ and $L > \ell$, as used in our construction): the reduction randomly permutes the order within each sample, and pads it with sufficiently many random dummy coordinates, before passing it to the PLDN solver.

5.3 Instantiation 2: Decoding via Concatenated Curves

A downside of the $\mathsf{Lift\text{-}RS}$-based construction from Theorem 5.2 is that it offers a limited level of asymptotic security. Since the rate of lifted RS codes provably approaches 1 only with smoothness parameter $t = O(\log\log\lambda)$ (Corollary 4.3), the corresponding sk-PIR scheme can be broken by adversaries that runs in quasi-polynomial time $\approx q^t = \lambda^{O(\log\log\lambda)}$ (see Section 6 for potential attacks).

As an alternative, we instantiate the sk-PIR construction with permuted RM codes and the local decoder based on concatenated RS, namely the permuted variant (Definition 5.2) of the locally decodable code $\mathsf{LDC}_{\mathsf{RM\text{-}Conc}} = (\mathsf{E}_{\mathsf{RM}}, \mathsf{Q}_{\mathsf{Conc}}, \mathsf{D}_{\mathsf{Conc}})$ defined in Fig. 4. Recall these codes are parameterized by the number of variables m, RM degree d, extension degree e, field size q and (weak) smoothness parameter t (cf. Theorem 4.1 and Corollary 4.2). In particular, we can take t to be k^c for constant $c > 0$, plausibly giving us security against quasi-polynomial adversaries.

Generally, we conjecture that $\mathsf{pLDC}_{\mathsf{RM\text{-}Conc}}$ is sk-PIR friendly whenever q^t is superpolynomial. This is in particular the case in the constant-rate instantiation of Corollary 4.2.

Conjecture 5.3 (sk-PIR Friendliness of $\mathsf{LDC}_{\mathsf{RM\text{-}Conc}}$)**.** *The permuted RM code with concatenated-curve decoding $\mathsf{pLDC}_{\mathsf{RM\text{-}Conc}}$ over $\mathbb{F}_q$ with smoothness parameter t is sk-PIR friendly whenever $t = \omega(1)$ and $q = \lambda^{\Omega(1)}$.*

Despite the similarity, the above conjecture is new to this work and is naturally less established than the PLD/PLDN conjectures (Conjectures 5.1 and 5.2). However, some of the cryptanalysis in Section 6 applies also to Conjecture 5.3, providing us with evidence that it stands natural types of attacks.

With the instantiation of Corollary 4.2, we obtain a doubly efficient sk-PIR based on concatenated-curve decoding, where the storage rate is constant.

Theorem 5.3. *Assuming Conjecture 5.3, for any constant $\varepsilon > 0$, there exists a doubly-efficient sk-PIR scheme with communication and server work k^{ε} and storage rate $\Omega(1)$.*

6 Cryptanalysis

In this section, we discuss major cryptanalysis attack families against our conjectured hard problems: PLD (Definition 5.3), PLDN (Definition 5.4) and PLDN-Conc(Fig. 2 and Conjecture 5.3). By now there are several works that consider asymptotic cryptanalysis of the assumptions we consider in this work [BIPW17, CHR17, BHW19, BW21, BHMW21] and related assumptions [DKL09, BCH$^+$25, CIMR26, CGG$^+$26].

The goal of this section is to substantially extend known asymptotic cryptanalysis, deriving sharper asymptotic conjectures for the permuted-low degree curves distribution PLD and its noisy variant PLDN (Conjectures 5.1 and 5.2) and our new conjecture for the permuted concatenated curves distribution PLDN-Conc (Fig. 2 and Conjecture 5.3). In addition, we provide concrete cryptanalysis, enabling us to derive concrete security evaluation of these assumptions (Conjectures 6.1 and 6.2 below).

Concrete Security Evaluation. We work with a standard notion of concrete security. We say that an assumption/scheme is concretely (T, S)-secure if any adversary that observes at most S samples from its oracle and runs in time at most T cannot win in the corresponding security game with advantage greater than $1/e$ (this choice is arbitrary, and one can work with any small constant here). We say that an assumption/scheme is $(T, *)$-secure if it is (T, S)-secure for any S. (In any case, the runtime T constitutes an upper bound on the number of samples S.)

Accordingly, we say that a scheme provides t *bits of security* given S samples, or given unbounded samples, if it is $(2^t, S)$-secure, resp. $(2^t, *)$-secure.

Based on the cryptanalysis in this section, we identify three classes of effective attacks against PLDN: The birthday attack (Section 6.2.2), pairwise intersection attacks (Section 6.2.5) and attacks based on detecting low algebraic rank (i.e. attacks against LSN problems, Sections 6.1.2 and 6.1.3).

The estimated complexity of the statistical attacks matches their sample complexity. Once the required number of samples is observed, distinguishing becomes computationally straight-forward. On the other hand, we are not aware of any statistical adversary that can achieve advantage $1/e$ using similar attacks with comparable runtime but fewer samples. On the other hand, the complexity of algebraic attack is not limited by their sample complexity and one can compensate fewer samples with larger runtime. Consequently, to break any of the assumptions, we conjecture that an adversary requires at least as many samples as needed by the best statistical attacks we identify and, when this is not met, then at least as much runtime as required by the best algebraic attacks we are aware of. Our conjectures are derived by a conservative estimate of the complexity of these attacks, on which we elaborate in the following subsections.

Conjecture 6.1 (Concrete Hardness of PLDN)**.** *For any integers $m \geq 2$ and $L \geq \ell > t > 2$, and prime power $q \geq \ell$, the (m, t, ℓ, L)-PLDN$_q$ over $\mathbb{F}_q$ problem is concretely (T, S)-secure where:*

$$S = \min\left(q^{m\cdot t/2-1}, t^{t^2-1}q^{m-2}\right) \qquad T = \min\left(\binom{\max(L, q-1)}{\ell}, q^{mL}\right)$$

and $(\min(S, T), *)$-secure in the unbounded-samples setting.

Conjecture 6.2 (Concrete Hardness of PLDN-Conc)**.** *For any integers $m, e \geq 2$, $t > 2$, $L \geq \ell = sr$, and prime power q such that $s \leq q^e - 1$, $r \leq q - 1$, the (m, t, e, s, r, L)-PLDN-Conc$_q$ over $\mathbb{F}_q$ problem is concretely (T, S)-secure where:*

$$S = \min\left(q^{e(m\cdot t/2-1)}, \quad t^{t^2-1}\cdot q^{e(m-2)}\right) \quad T = \min\left(\binom{\max(L, q^e-1)}{s}\cdot\binom{\max(L, q-1)}{r}^s, \quad q^{mL}\right)$$

and $(\min(S,T),*)$*-secure in the unbounded-samples setting.*

6.1 Algebraic Attacks

We describe a few ways in which one could exploit the algebraic structure present in the PLDN samples. Some attacks do not apply because of the safeguards and some take a long time.

6.1.1 McEliece-Inspired Attacks in the Absence of π

It is tempting to observe similarity between the instance generated by PLD_π (Definition 5.3) and the public-key of a McEliece encryption scheme with a Reed-Solomon code. After all, one picks a degree t parameteric curve ψ and evaluates it on random points $\mathbf{V} = (\psi(z_1), \ldots, \psi(z_\ell))$ in step 3, obtaining a Reed-Solomon codeword (Definition 3.3), up to the permutation π which acts as permutation over the Reed-Solomon alphabet. McEliece with Reed-Solmon code has been extensively studied (see [SS92, CGG$^+$13, Wie06] for example). We recall some major strategies to distinguish such an instance from random.

This part of the instance, without the alphabet permutation π is completely indistinguishable from random. Each row of $\mathbf{V}$ is in a hidden Reed-Solomon code of degree t and block length ℓ. In the setting, when $m \geq t$ and $\ell > t$, the Sidelenikov-Shekstov attack can recover the evaluation points as well as the polynomials up to an affine-relation [SS92]. Furthermore, the runtime of the attack is polynomial. The attack would also apply when $m < t$ and ℓ is sufficiently larger than t. To do this, one can use the Schur product trick. Recall that for two vectors $\mathbf{v}, \mathbf{w} \in \mathbb{F}_q^n$, the Schur product $\mathbf{v} \circ \mathbf{w} = (v_1 \cdot w_1, \ldots v_n \cdot w_n)$. The key property is that the Schur product of a two codewords (corresponding to polynomial g_1 and g_2) of a Reed-Solomon code with degree t and block-length ℓ is contained inside the Reed-Solomon code with same evaluation points but degree $2 \cdot t$ (and the resulting polynomial is simply $g_1 \cdot g_2$). As a result, from $\mathbf{V}$ one can create an expanded matrix $\mathbf{V}'$ with $\binom{m}{2} = \Omega(m^2)$ rows corresponding to Schur products of pairs of rows of $\mathbf{V}$. If the number of rows now exceed $2 \cdot t$, we can apply the Sidlenikov attack, if not, keep going and apply the Schur trick again to increase the number of rows further. Notice that the degree blow up with a degree d Schur product is just $d \cdot t$, but the number of rows increase as $\binom{m+d}{d}$. This attack can also be extended in the case when the curve evaluation part is embedded inside a random code by forming a column permutation f applied to $(\psi(z_1), \ldots, \psi(z_\ell), y_{\ell+1}, \ldots, y_L)$. The idea is that by doing repeated Schur attacks the columns corresponding to the Reed-Solomon code remain in a low-dimensional space, where as the random code component ends up being in a high dimensional space. One can then use Gaussian Elimination to find the column indices corresponding to the hidden RS code. This exact idea (in conjunction with another trick called Code Shortening) was used to cryptanalyze Wiescherbrink's McEliece variant [Wie06] in the attack [CGG$^+$13].

The attack described above therefore makes it critical that the permutation π is applied. If π is randomly chosen, it is not clear at all how one would apply the attack above. In fact, the original motivation of using π in [BIPW17, CHR17] has exactly been this attack.

6.1.2 Linear-Algebraic Attacks given Fixed Evaluation Points

From the previous attack, it is clear that there is no hope for security if π is leaked. One approach using which it might be plausible to learn π exploits the situation where there is no entropy in the set of evaluation points $z_1, \ldots, z_\ell$. A simple special case is when the set contains the entire curve, namely the case $\ell = q - 1$ of the *noiseless* assumption PLD (Definition 5.3).

LPN-style Attacks for Learning π. We start with the fact that

$$\sum_i \psi(z_i) = \mathbf{0}, \tag{10}$$

for all degree $t < q-1$ curves ψ. Given such a linear relation is given by any curve ψ, one can hope to find π by solving a system of linear equations as follows. We define q^m variables X over $\mathbb{F}_{q^m}$. Every sample gives us an equation of the form

$$\sum_z X_{\pi(\psi(z))} = \mathbf{0}. \tag{11}$$

corresponding to a new curve ψ. By collecting sufficiently many samples corresponding to simple client request pattern (e.g. a constant sequence of a repeated request), we can hope to solve the system to obtain the solution $X_c = \pi^{-1}(c)$ (possibly after guessing a few values).

When $\ell = q-1$, PLD is therefore not hard. The attack further extends to the case where $\ell < q-1$ and $z_1, \ldots, z_\ell$ are fixed. Any fixed choice of $z_1, \ldots, z_\ell$ defines a linear dependency over the $\psi(z_i)$, which is characterized by the corresponding Lagrange interpolation coefficients.

We instead conjecture hardness in the presence of noise. The noise either comes in the form of randomizing the evaluation points, sampling only a subset of $\mathbb{F}_q^*$ of size $\ell < q-1$ and creating entropy in the linear relation, and/or by sampling $L - \ell$ additional random values, making it a PLDN instance (Definition 5.4).

When $\max(L, q-1)$ is larger than ℓ but not by much, one may hope to guess the linear relation with some probability, obtaining a noisy system of linear equations resembling an instance of the learning parity with noise (LPN) problem. If one could solve an LPN system with q^m variables and error $\varepsilon = 1 - p$, then one can still apply the attack. However, whenever $\ell < \max(L, q-1)$ in our setting, ε is sufficiently large to deem any known generic attack against LPN completely ineffective.

LSN-style Rank Attacks. There is an even more direct attack that exploits the linear dependency in PLD samples with $\ell = q-1$, and at the same time effectively extends to $\ell < q-1$ or PLDN when the choice of parameters does not give a sufficiently low error rate ε.

Underlying Eq. (10) is the fact that, for any polynomial P of degree less than $q-1$, the sum $\sum_{z \in \mathbb{F}_q} P(z)$ is 0. Indeed, this is the reason why Reed-Muller is locally decodable over low-degree curves: The codeword itself is a low-degree polynomial and, therefore, its composition with a low-degree curve is also a low-degree polynomial and hence its evaluations over the curve exhibit a linear relation which is used for decoding. An analogous relation underlies concatenated-curve decoding and, in fact, any locally decodable code, as we discuss hereby in Section 6.1.3. For now, let us focus on the standard Reed-Muller case.

For any curve ψ, let $I_\psi \in (\mathbb{F}_q)^{\mathbb{F}_{q^m}}$ denote the indicator vector of the evaluation points $(\psi(z))_{z \in \mathbb{F}_q}$: we assign value c to $I_\psi(z)$ if w appears c times in $(\psi(z))$. Then, by the above, any I_ψ is orthogonal to any Reed-Muller codewords and, therefore, the set $\{I_\psi\}$ is contained in the dual Reed-Muller code and therefore has algebraic rank $k^\perp := q^m - k$, where $k \approx q^m/m!t^m$ is the (primal) Reed-Muller dimension. Applying a permutation over the code has no effect on this algebraic structure, which still holds over the indicator vectors of the permuted query sets.

This suggests the following rank attack, first observed in [BW21, BHMW21], for distinguishing between PLD samples corresponding to a sequence of a repeated client queries and samples corresponding to a sequence of uniformly random client queries. The attack simply computes the rank of $n := q^m$ such queries and outputs 0 if the rank is at most $k^\perp$ and 1 otherwise. In the former case, when the client's query is repeated, the indicator vectors of the punctured curves all fall in

the same coset of the dual RM code that is defined by the client's query. Therefore, they will have rank at most $k^\perp$. In the latter case, the rank is expected to be close to full with high probability.

When $\ell < \max(L, q-1)$, the linear relation that a given sample satisfies is hidden and could be guessed with probability $p \approx 1/\binom{\max(L,q-1)}{\ell}$. Randomly guessing the correct linear relation for every sample, the attack obtains a collection of samples where p-fraction of the samples come from a low-rank subspace. This becomes a *learning subspace of noise* (LSN) problem with *mixture noise*, which is a mixture learning problem where every sample comes from a hidden low-rank subspace with probability p and is random noise with probability $\varepsilon = 1 - p$. LSN with mixture noise was first considered in the cryptographic context in [DKL09] and more recently in [CIMR26].

When $p > k^\perp/n$, the mixture noise is still not sufficient to hide the low-rank structure in the samples; Indeed, out of n samples, more than $p \cdot n > k^\perp$ sample are expected to come from a rank-$k^\perp$ subspace and rank deficiency is still observed. Such an attack can be further generalized via tensoring to work against any value of p, requiring n^d samples, where $d \approx \log(1/p)/\log(n/k^\perp)$ [DKL09, CIMR26]. A conservative concrete lower bound for the sample complexity of these attacks, that we use in Conjectures 6.1 and 6.2 is therefore $1/p$.

A similar situation appears in permuted samples of random concatenated curves, underlying our sk-DEPIR construction based on concatenated-curve decoding. In that distribution, described in Fig. 2, the random choice of evaluation points appears both in the "outer" curve, where we choose random $z_1, \ldots, z_s \in \mathbb{F}_{q^e}^*$, and in each "inner" curve, where we choose random $w_{j,1}, \ldots, w_{j,r} \in \mathbb{F}_q$. Just as in PLDN, when the evaluation points are fixed, the samples exhibit a known linear dependency which can be used to break our conjecture in the absence of noise. This is in particular the case when s and r are set to their maximal possible values of $q^e - 1$ and, respectively, q, or are only far from them by a constant (in which case there is no sufficient entropy in the linear dependency). Here, a bound on the "LSN error" is obtained by requiring the adversary to guess the full concatenated evaluation pattern. Since a sample contains $\ell = sr$ structured points among L, s outer evaluations among $q^e - 1$, and r inner evaluations among $q - 1$ for each sampled outer point, the probability that a fixed guessed relation aligns with the sampled coordinates can be roughly (and conservatively) bounded by $p \lesssim 1/\binom{\max(L,q^e-1)}{s}\binom{\max(L,q-1)}{r}^s$.

In the next section, we further examine PLD and PLDN as special cases of LSN and, in particular, formalize them as such. This allows us to articulate the differences compared to other variants of LSN and to extend existing and future cryptanalytic work against LSN to the assumptions in this work, including a different variant of the above rank attack from [BCH+25].

6.1.3 The Learning Subspace with Noise Perspective

The above attack exploits the linear structure in permuted low-degree samples corresponding to a fixed set of evaluation points. We observe that the linear structure is not unique to curves. which are coming from a RM local decoder in the sk-PIR construction, but is rather found in the queries of any decoder of a linear code that we may use.

A (local) decoder of a linear code necessarily computes a linear function over the codeword it is applied to. Such decoder takes the following general structure: On input $i \in [n]$, the query algorithm Q samples a linear function $f : \mathbb{F}^n \to \mathbb{F}$, where $f(\mathbf{c}) := \sum_j f_j \cdot \mathbf{c}(j) = \mathbf{x}(i)$ for any $\mathbf{c} = \mathsf{E}(\mathbf{x})$, and outputs the query set $Q = \mathrm{Supp}(f) = \{j \mid f_j \neq 0\}$. The decoding algorithm D takes f as auxiliary information and computes it on its input $\mathbf{c}_Q$ to obtain $f(\mathbf{c})$. When the associated encoding function is systematic, $f(\mathbf{c}) = \mathbf{x}(i)$ holds only when f comes from the coset $C^\perp + \mathbf{u}_i$, where $\mathbf{u}_i$ is the i^{th} unit vector.

The query sets of a decoder, then, are the support sets of vectors coming from cosets of the dual code $C^\perp$. One may think of the $\mathsf{PLD}_\pi(i)$ and $\mathsf{PLDN}_\pi(i)$ distributions, then, as instances of

the following more general distribution: sample random vectors from the coset $C^\perp + \mathbf{u}_i$, where $C^\perp$ is a random hidden linear code, and obfuscate them with noise.

In PLD and PLDN, C is a permuted Reed-Muller code and $C^\perp$ is its dual, a permuted Reed-Muller code itself. The random distribution over codes is then generated by applying a random permutation over a fixed family of LDCs. There is no reason to believe, however, that this is the only way by which one can meaningfully sample hidden random LDCs.

The noise in PLD and PLDN comes in two different forms: The first form of noise is reflected by the fact that samples contain the support sets of the decoding functions and not the underlying coefficients (the f_j above) – essentially erasing all information of the coset vector elements except whether they are zero or non-zero. The second form of noise is special to PLDN, where additional random non-zero coordinates are added – this resembles the Hamming noise in LPN except here only zero coordinates can be "flipped" (non-zero coordinates are never made zero).

The first type of noise is meaningful only when there is entropy in the coefficients to begin with. If there is none, e.g. when the evaluation points are fixed, the distinguishing problem becomes easy as we saw in Section 6.1.2. In such a case, the second type of noise is necessary.

This general perspective captures the hardness underlying the sk-PIR based on concatenated-curve decoding as well (Conjecture 5.3), thus casting all of our distinguishing problems as *learning subspace with noise* (LSN) problems [DKL09, CIMR26]. It unifies the doubly efficient sk-PIR constructions of [BIPW17, CHR17] and this work with the sk-PIR constructions from [CIMR26, BCH+25] under one framework.

Recall the general LSN template that was used in [CIMR26] to build secret-key PIR: The assumption asserts that samples $\mathbf{c}_j \leftarrow C$ from a secret linear code $C \subset \mathbb{F}^n$, of dimension $k < n$, are indistinguishable from random vectors when altered with noise (notice we switched notation, from viewing the samples as coming from a dual code $C^\perp$ to coming from a primal code C). The general form of noise considered in [CIMR26] is that of hiding any $\mathbf{c}_j$ in a random *affine subspace*, i.e. $\mathbf{c}_j + E_j$ where E_j is a linear subspace of $\mathbb{F}^n$ sampled independently according to some distribution. [CIMR26] instantiates the framework with different choices of distributions over the secret code C and the subspace E_j to obtain different LSN assumptions (e.g. Basic LSN [DKL09] and Split LSN) that give sk-PIR protocols with different features.

Definition 6.1 (General LSN [CIMR26])**.** *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:*

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

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

Building on this general LSN framework, [BCH+25] proposes a variant called 1-Dimensional Split LSN (1D-SLSN) to build efficient encrypted matrix-vector product (EMVP), which can be viewed as generalization of sk-PIR. In 1D-SLSN, the samples are hidden inside a *linear* subspace that spans them, namely E_j such that $\mathbf{c}_j \in E_j$. However, in contrast to the above, the samples come from a coset of C, i.e. $\mathbf{c}_j + \Delta_j$, for random $\mathbf{c}_j \leftarrow C$ and adversarially chosen Δ_j that model the client's queries in the corresponding sk-PIR protocol. (The adversarial choice of queries is accommodated in [CIMR26] by the fact that, with affine-subspace noise, pseudorandomness of noisy codewords is equivalent to pseudorandomness of noisy coset vectors.)

Local LSN. In our doubly-efficient sk-PIR constructions, the client queries define linear functions that are local, namely that touch a small number of elements in the database. Correspondingly, when we cast the underlying hardness assumptions (Conjectures 5.1 to 5.3) as LSN-type assumptions, we get variants of LSN where the samples have low Hamming weight. Such samples are obtained by sampling a low-weight codeword $\mathbf{c}_j$ from a suitable code C, shifting it by a low-weight vector Δ_j, and hiding $\mathbf{c}_j + \Delta_j$ in a random low-weight linear subspace E_j, namely a subspace that is supported by a small subset of coordinates and therefore spans low-weight vectors only. Jumping ahead, in our variants, E_j are subspaces of the form $E_j = \{\mathbf{v} \mid \mathrm{Supp}(\mathbf{v}) \subseteq T_j\}$ where $T_j \subset [n]$ is a small subset of coordinates such that $\mathrm{Supp}(\mathbf{c}_j + \Delta_j) \subseteq T_j$.

This requires few modifications to the LSN framework: First, we can no longer argue indistinguishability from uniformly random vectors, but rather from an appropriate distribution over low-weight vectors.

Second, note that low-weight samples cannot possibly hide arbitrary shifts Δ_j that do not necessarily have high weight. We focus on the special case where the Δ_j are unit vectors. This is sufficient for sk-PIR, where the client queries may be represented by unit vectors whenever the database encoding is systematic (which is the case in all of our constructions).

Further, for noisy samples $E_j \ni \mathbf{c}_j + \Delta_j$, where $\Delta_j = \mathbf{u}_i$ is the i^{th} unit vector, to hide i, the codeword $\mathbf{c}_j$ must be sampled as a function of i: If we sample a low-weight $\mathbf{c}_j$ independently in i then it is supported by the i^{th} coordinate only with small probability. This means that $\mathbf{c}_j + \mathbf{u}_i$, and therefore E_j, are supported by the i^{th} coordinate with high probability and that i is revealed by E_j.

We therefore consider a PPT sampler S that takes as input the description of the secret code C and an index i, and outputs a coset vector $\mathbf{c} \in C - \mathbf{u}_i$, where $\mathbf{u}_i$ is the i^{th} unit vector (for simplicity, we switch to denote the coset vector by $\mathbf{c}$). We note that we allow arbitrary (polynomial-size) description of the code C. In particular, it may contain any auxiliary information generated at the sampling of C and we assume it contains the security parameter λ. Further, note that S does not take as input the adversary's choice but rather its re-labeling under a uniformly random permutation sampled independently. (Put differently, the adversary's choice may affect the distribution only by the pattern of repetitions in the sequence.) Clearly, this only makes distinguishing harder, given all other parameters are fixed. In the parameter regimes we consider in this work, the re-labeling is necessary and without it there exists a polynomial-time distinguisher. (This is since we consider high-rate codes and curves of low degree, which overall imply that a large fraction of the coordinates are expected to fall in the systematic part and, if these are not permuted, an attacker can interpolate.)

Overall, we may cast the hardness assumptions in this work as instances of the following general formulation of a *local LSN* problem.

Definition 6.2 (Local LSN)**.** *Let $\mathcal{C}(1^\lambda, 1^k, 1^n)$ be a PPT algorithm that outputs a description of a k-dimensional linear code $C \subset \mathbb{F}^n$ and let $S(C, i)$ be a PPT algorithm that on inputs $C \in \mathcal{C}(1^\lambda, 1^k, 1^n)$ and $i \in [n]$ samples a coset vector $\mathbf{c} \in C - \mathbf{u}_i$. For dimension parameters $k := k(\lambda)$ and $n := n(\lambda)$, locality parameter $\ell := \ell(\lambda)$ and $L := L(\lambda) \geq L$, the $(k, n, \ell, L, \mathcal{C}, S)$-$\mathsf{LocalLSN}$ assumption asserts that for secret $C \leftarrow \mathcal{C}(1^\lambda, 1^{k(\lambda)}, 1^{n(\lambda)})$ and for any $i_1, \ldots, i_m \in [n]$ where $m := m(\lambda)$ is a polynomial, we have:*

$$(\mathrm{Supp}(\mathbf{c}_j) \ \cup \ D_j \ )_{j \in [m]} \approx_c (R_j)_{j \in [m]},$$

where for each j we take independent samples $\mathbf{c}_j \leftarrow S(C, \pi(i_j))$, D_j is a uniform subset of $[n] \setminus \mathrm{Supp}(\mathbf{c}_j)$ of size $L - |\mathrm{Supp}(\mathbf{c}_j)|$ and R_j is a uniform subset of $[n]$ of size L.

Let $q := q(\lambda)$, $m := m(\lambda)$ and $d := d(\lambda)$ be polynomially bounded and let $\mathsf{RM} := \mathsf{RM}_q(m, d)$. Let $\mathsf{RM}^\perp$ be the dual code of RM, a Reed-Muller code itself, and let $\mathcal{C}_{\mathsf{RM}}$ denote the distribution

over permuted $\mathsf{RM}^{\perp}$ codes, namely the distribution that outputs the code $\mathsf{RM}^{\perp}$ permuted by a uniformly random permutation π (akin to Definition 5.2). Let S_{RS} be the algorithm that given $C \in \mathcal{C}_{\mathsf{RS}}$ and $i \in [n]$ samples a codeword $c \in C - \mathbf{u}_i$ as follows: Samples $(Q, \mathsf{aux}) \leftarrow \mathsf{Q}_{\mathsf{RS}}(i)$ and outputs the vector $\mathbf{c} \in \mathbb{F}^{q^m}$ where: $\mathbf{c}_j = 0$ for $j : \pi(j) \notin Q$ and $\mathbf{c}' = (\mathbf{c}_j)_{\pi(j) \in Q} \in \mathbb{F}^{\leq \ell}$ is the linear function applied by $\mathsf{D}_{\mathsf{RS}}(i, \cdot, \mathsf{aux})$ which satisfies $(\mathbf{c}')^{\intercal}(\mathsf{E}_{\mathsf{RM}}(\mathbf{x})_j)_{\pi(j) \in Q} = \mathbf{x}_i$ for all $\mathbf{x}$, and therefore $\mathbf{c} \in C - \mathbf{u}_i$ (recall the decoder of a linear LDC is necessarily linear; in the RM case, $\mathbf{c}'$ consists of Lagrange interpolation coefficients).

Observe that $\mathsf{LDC}_{\mathsf{RM\text{-}RS}}$ with the above parameters is sk-PIR friendly with $L > \ell$ whenever $(\binom{d+m}{m}, q^m, \ell = dt + 1, L, \mathcal{C}_{\mathsf{RS}}, S_{\mathsf{RS}})$-$\mathsf{LocalLSN}$ holds. Similar implications hold for $\mathsf{LDC}_{\mathsf{RM\text{-}Conc}}$ and $\mathsf{LDC}_{\mathsf{Lift\text{-}RS}}$ with respect to appropriate choices of code and sampler.

Further, notice that the above local LSN problem is at least as hard as the corresponding PLD/PLDN problem. Equivalence does not hold since in the latter the samples consist of lists of coordinates (multisets).

A variant of the rank attack against general LSN problems from [BCH+25], in the spirit of [AG11], uses the fact that LSN samples exhibit a degree-r dependency, where r is the dimension of the samples, namely the linear subspaces E_j in which the codewords are hidden. The general attack has sample complexity roughly n^r. While structured variants of the LSN assumptions (e.g. 1D-Split-LSN [BCH+25]) may allow for optimizations, the attack remains infeasible for large values of r, with complexity growing exponentially. In our variants of local LSN, we have $r = L$, hence the last term in Conjectures 6.1 and 6.2. [BCH+25] also shows that such generic algebraic attacks cannot do better. This indicates that improved algebraic attacks against local LSN must rely on the local structure and other properties of the underlying codes.

6.2 Statistical Attacks

We next discuss potential statistical attacks that attempt to exploit the combinatorial structure of permuted low-degree curves over $\mathbb{F}_q^m$. In contrast to the algebraic structure which is completely disrupted in the presence of noise (in the form of entropy in the evaluation points or additional random coordinates), attacks that look for combinatorial properties of the samples are not always affected by such noise, and when they are, they can be often modified to account for it. Thus, when analyzing such attacks, we focus on the noiseless PLD problem (Definition 5.3) with $\ell = q-1$, where the sample contain all points on the curve except the evaluation at 0.

To extend our cryptanalysis of PLD to the concatenated-curve distribution underlying Conjecture 5.3, namely $\mathsf{PLDN\text{-}Conc}$ (Fig. 2), we most look on how applying an additional "inner-curve encoding" affects the combinatorial structure of the samples. Recall from Fig. 2 that a concatenated-curve sample is generated by first sampling an outer degree-t curve

$$\psi : \mathbb{F}_{q^e} \to \mathbb{F}_{q^e}^m$$

conditioned on a planted $\psi(0)$ value, and then expanding sampled outer points through an inner Reed-Solomon evaluation over $\mathbb{F}_q$. That is, instead of seeing a random permutation of curve points over $\mathbb{F}_{q^e}$, an adversary would now observe random encodings of its $\mathbb{F}_{q^e}$-points, and the permutation is applied individually over the encoding symbols that are over $\mathbb{F}_q$.

On the one hand, concatenation seems to reduce combinatorial structure: For example, if two outer curves intersect at some point over $\mathbb{F}_{q^e}$, the two inner encodings of this point in the corresponding observed outcomes intersect only with low probability (that depends on the choice of r and q) due to the randomness in the encoding. On the other hand, concatenation potentially introduce additional structure: In PLD, two distinct points map to two uniformly random distinct

points under the permutation. In the concatenated variant, correlations between the two points may be revealed through their encodings, as the permutation is applied individually over the encoding symbols. This being said, it is not clear how the adversary can tell which points in the concatenation belong to which outer symbols, beyond guessing.

We generally conjecture that, for statistical attacks, this inner expansion does not provide exploitable information beyond the outer curve structure, once the global permutation and the random ordering of evaluation points (fresh per sample) are applied. Explicitly, we believe that statistical attacks against PLDN-Conc should be no more effective than the corresponding attacks against PLD/PLDN with field size q^e and the same m and t. This should be viewed as a general heuristic, which we will make more explicit for the attacks we consider to be most effective, deriving Conjecture 6.2.

6.2.1 Preliminary Statistical Analysis

Whenever $q, m, t \in \mathbb{N}$ are implicit by context, we let Ψ denote the set of all parametric curves over $\mathbb{F}_q^m$ of degree at most t. For any $i \in \mathbb{F}_q^m$, we denote by Ψ_i the set of any such curve ψ satisfying $\psi(0) = i$. Additionally, for any curve ψ, we let

$$V(\psi) = (\ \psi(z)\ )_{z \in \mathbb{F}^*}, \tag{12}$$

and denote

$$\mathcal{V} = \{V(\psi) \mid \psi \in \Psi\} \quad \text{and} \quad \mathcal{V}_i = \{V(\psi) \mid \psi \in \Psi_i\}.$$

In particular, a subset $V \leftarrow \mathcal{V}_i$, that undergoes re-labeling under the permutation π, distributes like a sample from the oracle $\mathsf{PLD}_\pi(i)$ (Definition 5.3).

Proposition 6.1. *For any $t, m \in \mathbb{N}$ and $\alpha \in \mathbb{F}^m$, $|\mathcal{V}_\alpha| = |\mathcal{V}|/q^m$.*

Proof. The proposition follows from the fact that $|\mathcal{V}_i|$ is equal over all $i \in \mathbb{F}^m$ due to symmetry and that for any i, i', the sets $\mathcal{V}_i$ and $\mathcal{V}_{i'}$ are disjoint. □

In the following proposition, we give an upper bound on the cardinality of $\mathcal{V}$ and, therefore, of the distributions Rand and PLD_π (Definition 5.3).

Proposition 6.2. *For any $t, m \in \mathbb{N}$, it holds that*

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

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 t, which bounds the number of possible curves in Ψ. There are $q^{m(t+1)}$ such choices. Observe, however, that different choices of polynomials (i.e. curve) may produce equal value sets. For instance, any two curves $\psi = (\psi_1, \ldots, \psi_m)$ and $\psi' = (\psi'_1, \ldots, \psi'_m)$ where $\psi'_i = \psi_i \circ \varphi$, for a non-constant affine function φ over $\mathbb{F}_q$ (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 $\mathbb{F}_q$, 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.

Conjecture 6.3. *For any $t, m \in \mathbb{N}$, it holds that*

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

when $q \to \infty$.

Looking back at the proof of Proposition 6.2, 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 $\mathbb{F}_q$. While we are not aware of any complete rigorous analysis for the number of low-degree permutation 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 6.3, by empirical data as demonstrated in Fig. 6.

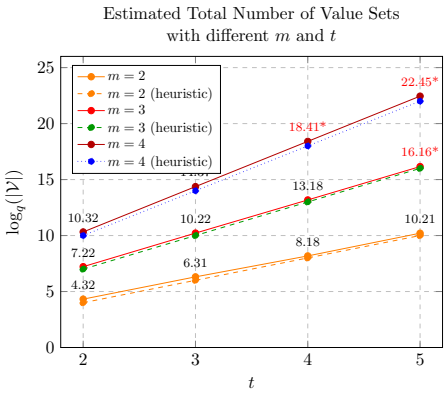

Figure 6: Estimated size of $\mathcal{V}$ for different choices of m and t. The experiments were conducted to mimic a birthday attack, i.e. to find a first repetition in random samples. By the birthday paradox, the total number of curves is estimated to be the square of number of samples required for a repetition. Each point represents an average of 1000 runs, and points marked with an asterisk (*) are extrapolated.

6.2.2 A Birthday Attack

An immediate consequence of Propositions 6.1 and 6.2 is a birthday-attack solver for PLD, which makes $\sqrt{|\mathcal{V}_i|} = q^{mt/2-1}$ queries and has constant advantage. More generally, one may obtain the following trade-off between the number of queries and advantage.

Proposition 6.3 (Birthday Attack)**.** *For any $\gamma \in (0, 1)$, there exists an algorithm that solves $(m, t, q-1)$-PLD with advantage $1 - \Theta(e^{-\gamma^2})$ by making $\gamma \cdot q^{mt/2-1}$ queries to its oracle.*

Assuming Conjecture 6.3, Proposition 6.3 captures the best trade-off that can be achieved by a birthday attack.

Conjecture 6.4 (Birthday Attack Concrete Lower Bound)**.** *A birthday attack against* PLD *requires* $\sqrt{2\ln(e/e-1)} \cdot q^{mt/2-1} \approx 0.96 \cdot q^{mt/2-1}$ *samples to achieve advantage* $1/e$*.*

While Conjecture 6.3 is a relatively solid heuristic, we stress that one may easily obtain an unconditional, albeit weaker, lower bound on the cardinality of $\mathcal{V}$, which in turn implies an unconditional lower bound on the complexity of a birthday attack against PLD.

Recall the reason Proposition 6.2 is not strictly tight is that different curves may give identical value sets due to low-degree permutation polynomials. To avoid quotienting by arbitrary permutations of $\mathbb{F}_q$, we may restrict to the subset of curves where the first coordinate is the identity, i.e. $\psi = (\psi_1, \ldots, \psi_m)$ such that $\psi_1(x) = x$. Under such restriction, distinct choices of the remaining $m-1$ polynomials give distinct value sets. Hence, the subset has cardinality at least $q^{(m-1)(t+1)}$ and we obtain the following bound.

Proposition 6.4 (Birthday Attack Crude Lower Bound)**.** *For any* $t, m \in \mathbb{N}$*, it holds that*

$$|\mathcal{V}| \geq q^{(m-1)(t+1)}.$$

In particular, a birthday attack against PLD *requires* $q^{\Omega(mt)}$ *samples to achieve constant advantage.*

Concatenated curves. Our lower bounds from Conjecture 6.4 and Proposition 6.4 extend directly to lower bounds on the complexity of birthday attacks against permuted concatenated curves, i.e. PLDN-Conc (Fig. 2), that are at least as a conservative. Let $Q = q^e$ be the size of the extension field used in PLDN-Conc. At the outer-curve level, PLDN-Conc samples degree-t curves conditioned on $\psi(0) = i$, producing outer values from $\mathcal{V}_i$. Applying our analysis from above with q replaced by Q, the number of possible outer value sets for a fixed i is about Q^{mt-2} (Conjecture 6.3) and at least $Q^{(m-1)(t+1)}$ (Proposition 6.4).

The inner-curve encoding does not reduce the size of this support by the following simple argument: fix any canonical choice of $r \geq e$ distinct inner evaluation points $w_1, \ldots, w_r$. Then the inner map

$$\phi_{w_1,\ldots,w_r} : \mathbb{F}_{q^e} \to \mathbb{F}_q^r$$

is injective, since a degree-$< e$ polynomial is determined by its values at e distinct points. Thus, distinct outer value sets remain distinct after this fixed inner expansion. Since the actual PLDN-Conc distribution ranges over all choices of the w's, its support is at least as large as this fixed-w subfamily.

Thus, Proposition 6.3 and Conjecture 6.4 extend directly to PLDN-Conc by applying the same analysis to curves over the field $\mathbb{F}_Q$, i.e., by replacing q in the bounds with Q.

Note that the above argument does not rule out accidental overlaps caused by the fact that the w-values are drawn freshly at every sample, but such overlaps can only affect the tightness of our bounds and not their correctness.

6.2.3 An Attack on the $t = 1$ Case

Next, we show that PLD with $t = 1$ is susceptible to a simple statistical attack. In particular, the attack is able to distinguish between two samples from $\mathsf{PLD}_\pi(i)$ corresponding to the same i and two samples corresponding to different inputs, namely $\mathsf{PLD}_\pi(i)$ and $\mathsf{PLD}_\pi(i')$. This directly translates to an attack against a sk-PIR protocol based on PLD with $t = 1$ that can tell whether two PIR queries correspond to a repeated client's index.

Lemma 6.1. *Assume Conjecture 6.3 holds*[6]*. There exists an algorithm solves* $(m, 1, q-1)$-PLD*, with advantage* $\Omega(1/q^{m-1})$ *in time and space linear in* q*, by making only two queries to its oracle.*

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 $\mathsf{PLD}_\pi(i)$, for any $i \in \mathbb{F}_q^m$, never intersect, whereas two samples corresponding to two uniformly random values $i, i' \leftarrow \mathbb{F}_q^m$ do with good probability.

Claim 6.1. *For any pair of degree-1 curves* ψ, ψ' *over* $\mathbb{F}_q^m$*, it holds that either* $V(\psi) = V(\psi')$ *or* $|V(\psi) \cap V(\psi')| < 2$*.*

Proof. Denote by $V := V(\psi)$ and $V' := V(\psi')$ the sets of points corresponding to the two curves.

Assume there exist two distinct $y_1, y_2 \in V \cap V'$ and let

$$\begin{aligned} y_1 &= \psi(z_1) = \psi'(z_3) \\ \text{and } y_2 &= \psi(z_2) = \psi'(z_4), \end{aligned}$$

for some $z_1, z_2, z_3, z_4 \in \mathbb{F}_q$ such that $z_1 \neq z_2$ and $z_3 \neq z_4$. Then, by linearity of ψ, ψ', we have for, any $\delta \in \mathbb{F}_q$,

$$\begin{aligned} &\psi(z_1 + \delta \cdot (z_2 - z_1)) \\ &= \psi(z_1) + \delta \cdot (\psi(z_2) - \psi(z_1)) \\ &= \psi'(z_3) + \delta \cdot (\psi'(z_4) - \psi'(z_3)) \\ &= \psi'(z_3 + \delta \cdot (z_4 - z_3)) \end{aligned}$$

For $z_1 \neq z_2$, the function $\delta \mapsto z_1 + \delta \cdot (z_2 - z_1)$ is bijective over $\mathbb{F}_q$, which means that $z_1 + \delta \cdot (z_2 - z_1)$ enumerates all values in $\mathbb{F}_q$ when δ goes over $\mathbb{F}_q$. The same holds for the function $\delta \mapsto z_3 + \delta \cdot (z_4 - z_3)$. Thus, we conclude that V and V' contain the same set of points in $\mathbb{F}_q^m$. □

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

$$\Pr_{V, V' \leftarrow \mathsf{PLD}_\pi(i)}[0 < |V \cap V'| < q] = 0. \tag{13}$$

It remains to lower bound the probability of $\mathcal{A}$ outputting 1 in the case where V and V' are sampled from PLD_π for uniform values of i, namely from $\mathcal{V}$. Let ψ, ψ' be two uniformly random degree-1 curves over $\mathbb{F}_q^m$. Due to Claim 6.1, we may calculate the probability that $V = V(\psi)$ and $V' = V(\psi')$ intersect at one point as follows (recall definition in Eq. (12))

$$\begin{aligned} &\Pr_{V, V'}[|V \cap V'| = 1] \\ &= \sum_{z \in \mathbb{F}_q^*} \Pr_{V, \psi'}[\psi'(z) \in V] - \Pr[V = V'] \\ &\geq (q-1)/q^m - 1/|\mathcal{V}| = \Omega(1/q^{m-1}), \end{aligned}$$

where the last equality is due to Conjecture 6.3. This, together with Eq. (13), completes the proof of Lemma 6.1.

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

6.2.4 Statistical Query Attacks

In [BHW19], it is shown that PLD is resilient against *statistical query (SQ) algorithms*. These are algorithms that do not receive direct samples from the PLD distribution but only the result of some apriori-chosen binary predicate $f(\cdot)$ applied over them. In fact, the lower bound applies to a simplified, so called *toy version*, of the problem, and shows that any SQ algorithm must make exponentially many queries to gain distinguishing advantage.

6.2.5 Pairwise Intersection Attacks

Next, we analyze attacks that look at *pairwise intersections* in samples from the PLD oracle. The attack template is as follows: The attacker observes N samples $V_1, \ldots, V_N \in \mathcal{V}$ and, for any possible intersection size $0 \leq s \leq q-1$, counts the number of pairs i, j such that $|V_i \cap V_j| = s$.

Due to symmetry, it is sufficient to evaluate the statistical difference between the case where all samples are taken uniformly at random from $\mathcal{V}_i$, i.e. correspond to a sequence of repeated client query i, and the case where all samples are uniform over $\mathcal{V}$, i.e. correspond to uniformly random sequence of client queries.

The case of $m = 2$. In two dimensions, curves are also hypersurfaces (have co-dimension 1). In such case, Bézout's theorem states that two coprime degree-t curves (i.e. two curves that do not share a common irreducible divisor; this holds with overwhelming probability for uniform curves) intersect at exactly t^2 points in the algebraic closure of $\mathbb{F}_q^m$, including multiplicities. This hints to the possibility of a distinguishing advantage: two random samples from $\mathcal{V}$ will typically have larger intersection size compared to two samples from $\mathcal{V}_i$, which correspond to two curves that already intersect at the omitted point (at zero). To analyze the advantage, however, we must understand how many of the intersection points over the algebraic closure (out of the t^2, resp. $t^2 - 1$) are $\mathbb{F}_q$-rational, namely fall in $\mathbb{F}_q^2$ and are actually visible to the adversary.

To that end, we invoke a standard heuristic from algebraic geometry concerning the behavior of the Frobenius endomorphism over sets in a general position (the so-called "Chebotarev/random Frobenius philosophy", cf. [KS99, Ser16]). The reason why we look at Frobenius is that the fixed points under the endomorphism are exactly all $\mathbb{F}_q$-rational points: $x^q = x$ for all $x \in \mathbb{F}_q$, giving us an algebraic characterization of the points visible in the samples.

Now, observe that the Frobenius endomorphism defines a permutation over the intersection of any two curves, since the intersection is closed under the map $(x, y) \mapsto (x^q, y^q)$. We assume that when the two curves are random, their intersection is in general position w.r.t. Frobenius, i.e. the elements in the intersection does not posses additional algebraic structure. This allows us to invoke the "random Frobenius" heuristic, postulating that the endomorphism over such sets behaves like a uniformly random permutation.

Let F_S denote the number of fixed points in a random permutation over S elements. Under the above heuristic, the distinguishing advantage from a single pair of samples boils down to the statistical distance between F_{t^2} and F_{t^2-1}, which by simple calculations may be bounded by

$$d_{\mathrm{TV}}(F_{t^2}, F_{t^2-1}) = \frac{2^{t^2-1}}{(t^2)!} \leq e^{-t^2 \ln t},$$

for any $t \geq 4$.

To obtain a constant distinguishing advantage, the attack requires exponentially in $t^2 \ln t$ many samples. Notably, however, the attack's complexity does not grow with q.

The case of $m > 2$. In three dimensions or more, curves do not coincide with hypersurfaces and Bézout's theorem no longer gives a sharp characterization of pairwise intersection sizes (rather it only gives an upper bound of t^m on the m-wise intersection size). A simple crude heuristic is that intersections restricted to the first two coordinates generate about $\exp(-t^2 \ln t)$ statistical distance, while each additional coordinate must also match and heuristically behaves like an independent constraint satisfied with probability $1/q$.

Conjecture 6.5 (Pairwise Intersections Concrete Lower Bound)**.** *Pairwise intersection size attacks against* PLD *require at least $e^{t^2 \ln t - 1} \cdot q^{m-2}$ samples to achieve advantage $1/e$.*

The above heuristic is useful to derive concrete security evaluation in practical parameter regimes. Asymptotically, however, it turns out to be too conservative when q grows much faster than t. In what follows, we show that the advantage gained by pairwise intersection size statistics decays exponentially in $\sqrt{t} \cdot \log(q)$, whenever $m > 2$.

For any pair of curves $\psi, \varphi : \mathbb{F}_q \to \mathbb{F}_q^m$, we define the bipartite graph $G_{\psi,\varphi}$ over vertices $\mathbb{F}_q^* \times \mathbb{F}_q^*$, where

$$(z, w) \in E(G_{\psi,\varphi}) \iff \psi(z) = \varphi(w).$$

Thus $G_{\psi,\varphi}$ records exactly which points of the two punctured domains give the same point in $\mathbb{F}_q^m$.

Any intersection statistic over pairs of samples $V_1, V_2 \in \mathcal{V}$ is a deterministic outcome of the graph $G_{\psi,\varphi}$, where ψ, φ are the degree-t curves underlying the two value sets (namely that satisfy $V_1 = V(\psi)$ and $V_2 = V(\varphi)$). Therefore, to bound the distinguishing advantage gained by pairwise intersection statistics, it suffices to bound the statistical distance between a graph $G_{\psi,\varphi}$ corresponding to two uniformly random curves ψ, φ, and the same graph when ψ, φ are sampled conditioned on $\psi(0) = \varphi(0) = 0^m$ (w.l.o.g.).

Lemma 6.2 (Pairwise Intersections Asymptotic Lower Bound)**.** *Let $m \geq 3$ and $3 \leq t \leq q - 1$ be such that $\log q \gg \sqrt{t} \log t/(m-2)$. Let G_{R} be the graph $G_{\psi,\varphi}$ obtained from i.i.d. $\psi, \varphi \leftarrow \Psi$, and let G_{P} be the graph $G_{\psi,\varphi}$ obtained from i.i.d. $\psi, \varphi \leftarrow \Psi_0$. Then,*

$$d_{\mathrm{TV}}(G_{\mathsf{R}}, G_{\mathsf{P}}) = q^{-\Omega((m-2)\sqrt{t})}.$$

Proof. By th $(t+1)$-wise independence of degree-t polynomial evaluations, namely the t-smoothness of the corresponding LDC, the two distributions agree on every pattern involving at most t points from each curve. That is, for any $F \subseteq \mathbb{F}_q^* \times \mathbb{F}_q^*$ of size at most $|F| \leq t$ it holds that

$$\Pr[F \subseteq E(G_{\mathsf{R}})] = \Pr[F \subseteq E(G_{\mathsf{P}})].$$

Next, we use the following generic bound on the statistical distance between subset distributions, which is implied whenever the distributions are identical up to some subset size. The proof is based on simple inclusion-exclusion and is provided in Appendix A.

Claim 6.2. *Let $X, Y \subseteq \Omega$ be random subsets of a finite set Ω. Suppose that*

$$\Pr[A \subseteq X] = \Pr[A \subseteq Y] \qquad \text{for every } A \subseteq \Omega,\ |A| \leq t.$$

Then

$$d_{\mathrm{TV}}(X, Y) \leq 2^t \left(\mathbb{E}\binom{|X|}{t+1} + \mathbb{E}\binom{|Y|}{t+1} \right).$$

Apply the claim with $\Omega = \mathbb{F}_q^* \times \mathbb{F}_q^*$, $X = E(G_{\mathsf{R}})$ and $Y = E(G_{\mathsf{P}})$. Since the containment probabilities agree for all edge sets of size at most t, we get

$$d_{\mathrm{TV}}(G_{\mathsf{R}}, G_{\mathsf{P}}) \le 2^t (M_{\mathsf{R}} + M_{\mathsf{P}}),$$

where, for $r = t + 1$,

$$M_{\mathsf{B}} = \mathbb{E}\binom{|E(G_{\mathsf{B}})|}{r}, \qquad \mathsf{B} \in \{\mathsf{R}, \mathsf{P}\}.$$

At this point, all contributions from edge patterns of size at most t have cancelled exactly. The task is therefore reduced to bounding the first non-cancelled factorial moment, namely the expected number of contained edge sets of size $r = t + 1$. Writing

$$M_{\mathsf{B}} = \sum_{\substack{F \subseteq \mathbb{F}_q^* \times \mathbb{F}_q^* \\ |F| = r}} \Pr[F \subseteq E(G_{\mathsf{B}})], \tag{14}$$

we proceed to bound the probability that a fixed r-edge pattern appears in a graph G_{B}, and then sum over all possible patterns.

Fix $F \subseteq \mathbb{F}_q^* \times \mathbb{F}_q^*$ with $|F| = r$, and view F as a bipartite graph after removing isolated vertices. Let $v(F)$ be the number of active vertices, let $c(F)$ be the number of connected components, and set

$$d(F) = v(F) - c(F).$$

A spanning forest of F has $d(F)$ edges. Keeping only $d_0(F) = \min(d(F), t)$ forest edges and using again the t-wise independence gives, for both $\mathsf{B} \in \{\mathsf{R}, \mathsf{P}\}$,

$$\Pr[F \subseteq E(G_{\mathsf{B}})] \le q^{-m \cdot d_0(F)}.$$

Indeed, a forest with $d_0 \le t$ edges imposes exactly d_0 independent equalities in $\mathbb{F}_q^m$.

Now, back to Eq. (14), we sum over all r-edge patterns. For fixed numbers $a, b \le r$ of non-isolated left and right vertices, the number of possible r-edge bipartite graphs is at most

$$\binom{ab}{r} \le \binom{r^2}{r} \le (er)^r,$$

and there are at most r^2 choices of (a, b). For any fixed choice of a and b, there are at most q^{a+b} embeddings into $\mathbb{F}_q^* \times \mathbb{F}_q^*$. Therefore

$$M_{\mathsf{B}} \le r^2 (er)^r \max_{|F| = r} q^{v(F) - m \cdot d_0(F)}.$$

To bound the above expression, we use the following two simple inequalities.

Claim 6.3. $d(F) \ge 2\sqrt{r} - 1$.

Proof of claim. Let $F_1, \ldots, F_c$ be the connected components of F, and let v_j and e_j denote the number of active (non-isolated) vertices and, respectively, edges in F_j. Since F_j is bipartite,

$$e_j \le \frac{v_j^2}{4}.$$

Therefore

$$r = \sum_{j=1}^{c} e_j \le \frac{1}{4} \sum_{j=1}^{c} v_j^2.$$

Write $d_j = v_j - 1$. Since each active component contains at least one edge, $d_j \geq 1$, and

$$d(F) = \sum_{j=1}^{c} d_j.$$

Moreover,

$$\sum_{j=1}^{c} v_j^2 = \sum_{j=1}^{c} (d_j + 1)^2 \leq \left(\sum_{j=1}^{c} d_j + 1 \right)^2 = (d(F) + 1)^2.$$

Hence

$$r \leq \frac{1}{4}(d(F) + 1)^2,$$

and so $d(F) \geq 2\sqrt{r} - 1$. □

Claim 6.4. *$v(F) \leq 2d(F)$.*

Proof of claim. The inequality follows immediately by definition as

$$d(F) = v(F) - c(F) = \sum_{j=1}^{c(F)} (v_j - 1) \geq c(F),$$

and $v(F) = d(F) + c(F)$. □

If $d(F) \leq t$, then $d_0(F) = d(F)$. By Claims 6.3 and 6.4,

$$q^{v(F) - m \cdot d_0(F)} \leq q^{-(m-2)d(F)} \leq q^{-(m-2)(2\sqrt{r}-1)}.$$

If $d(F) > t$, then $d_0(F) = t$. Since $v(F) \leq 2r = 2t + 2$ by definition,

$$q^{v(F) - m \cdot d_0(F)} \leq q^{2t+2-mt} = q^{2-(m-2)t} \leq q^{2-(m-2)(2\sqrt{r}-1)},$$

whenever $t \geq 3$. In either case, we conclude

$$M_{\mathsf{B}} \leq r^2 (er)^r q^{2-(m-2)(2\sqrt{r}-1)}.$$

Substituting this bound for both $\mathsf{B} = \mathsf{R}$ and $\mathsf{B} = \mathsf{P}$ gives

$$d_{\mathrm{TV}}(G_{\mathsf{R}}, G_{\mathsf{P}}) \leq r^2 (2er)^r q^{2-(m-2)(2\sqrt{r}-1)}.$$

Under the assumed lower bound on q, the distance may be asymptotically bounded by

$$d_{\mathrm{TV}}(G_{\mathsf{R}}, G_{\mathsf{P}}) \leq q^{-\Omega((m-2)\sqrt{t})}.$$

□

Concatenated curves. To reason about the resilience of PLDN-Conc (Fig. 2) against pairwise intersection statistics, we recall that the outer curve induces an ordinary PLD instance over $\mathbb{F}_{q^e}$. Pairwise intersections of the observed base-field samples, however, are not determined solely by outer intersections. Indeed, even two distinct outer symbols $a \neq b \in \mathbb{F}_{q^e}$ may yield overlapping inner evaluations under independently sampled inner evaluation sets. Thus, the PLD intersection analysis over the outer field does not formally imply the corresponding statement for PLDN-Conc. Nevertheless, it suggests the right structured contribution: outer intersections should behave as in PLD with q replaced by $Q = q^e$, while intersections not induced by an outer collision arise only from accidental overlaps in the inner RS evaluations. We heuristically treat these accidental inner overlaps as query-independent noise, contributing essentially the same distribution in the planted and random cases. Under this additional heuristic, the pairwise-intersection advantage against PLDN-Conc is bounded by the advantage against PLD with q replaced by Q. We leave further analysis to future work.

6.2.6 Local Distinguishers

We now reason about the resilience of our assumptions against *local distinguishers*. These are algorithms that collect local data over the samples, namely functions that depend on a small number of points in the samples.

We consider two natural interpretations of locality. Recall that a PLD sample consists of a list of values $Q = (y_1, \ldots, y_\ell)$.

First, we consider algorithms that look at t locations in the list $(y_{i_1}, \ldots, y_{i_t})$. By the t-smoothness of the curve-based decoder, any such t values distribute like uniform i.i.d. variables, regardless of the adversarial choice of the input to the PLD oracle. Therefore, PLD is resilient against t-local attacks *even in the absence of the random permutation* π, which makes distinguishing only harder.

Proposition 6.5 (Local Attacks against PLD)**.** *Adversaries against* PLD *that look at* $\leq t$ *locations in every sample cannot achieve non-zero advantage.*

While we cannot make an identical argument for our weakly smooth variant, the concatenated structure of our weakly smooth local decoder from Fig. 4 satisfies the following "on average" notion of (strong) smoothness: for $t' = o(\sqrt{t})$, $(y_{i_1}, \ldots, y_{i_{t'}})$ distribute like i.i.d. variables for *most* $i_1, \ldots, i_{t'}$, specifically for all but negligible fraction (that decays inverse-sub-exponentially in t). At a high level, this is because the sample consists of $s > t$ "blocks of dependency", namely the "small" degree-$(e-1)$ curves (cf. Fig. 3), and t' indices are i.i.d. unless two of them come from the same block. The chance of such a collision when $t' = o(\sqrt{t})$ is negligible.

Given the samples undergo a random re-ordering in the sk-PIR scheme, the best any t'-local attack can do is to leverage dependency within *random* t'-tuple of positions, which is by the above impossible even in our weakly smooth variant.

Proposition 6.6 (Local Attacks against Permuted Concatenated Cruves)**.** *Adversaries against* PLDN-Conc *that look at* $o(\sqrt{t})$ *locations in every sample cannot achieve non-negligible advantage.*

We stress that the above propositions give a lower bound on the statistical advantage gained by local adversaries. While higher locality necessarily produces noticeable statistical gap (due to simple entropy considerations), we are not aware of a computationally efficient distinguisher that can obtain noticeable advantage using higher locality.

A second notion of locality is the following: The algorithm is allowed to look at whether $y \in Q$, for a small number of values $y \in [n]$, say at most r. While smoothness does not directly apply,

we can still show that distinguisher's view over $\leq t$, resp. $o(\sqrt{t})$, such membership predicates is statistically close in the two distributions.

In Lemma 6.3, we prove that whenever $q \cdot t \ll n$, such membership predicates are close to be t-wise independent (resp. $o(\sqrt{t})$) as long as the original variables are t-wise independent (resp. $o(\sqrt{t})$) and have uniform marginals. As a consequence, Propositions 6.5 and 6.6 imply analogous lower bounds for adversaries that are local in the above sense.

First, we recall the following useful fact about t-wise independent variables from [GY20] then use it to prove the lemma.

Fact 6.1 ([GY20], Theorem 1.1)**.** *Let $X_1, \ldots, X_n \in [-1, 1]$ be t-wise independent random variables. Then, there exist constants $c_1 > 1$ and $1 > c_2 > 0$ such that*

$$\left| \mathbb{E}[\prod_{j=1}^{n} X_j] - \prod_{j=1}^{n} \mathbb{E}[X_j] \right| \leq \left(c_1 \cdot \sqrt{\sum_{j=1}^{n} \mathrm{Var}(X_j)} \right)^{c_2 t}.$$

.

Lemma 6.3. *Let $k < t \ll n$. Let $x_1, \ldots, x_\ell$ be t-wise independent random variables over $[n]$ with uniform marginals, and let $y_1, \ldots, y_n$ be the binary random variables taking $y_i = 1$ if and only if $x_j = i$ for some j.*

Let $u_1, \ldots, u_\ell \leftarrow [n]$ be uniform i.i.d., and let $w_1, \ldots, w_n$ be the binary random variables taking $w_i = 1$ if and only if $u_j = i$ for some j.

Then, for any set $I \subset [n]$ of size k, the marginal of y_I is ε close to the marginal of w_I, where

$$\varepsilon = 2^{2k} \cdot \left(\frac{\ell k}{n} \right)^{\Omega(t)}.$$

Proof. For any $S \subseteq [n]$ with $|S| = s \leq k$, define the event E_S where $y_i = 0$ for all i (equivalently, $x_j \notin S$ for all j). Let $X_j \in \{0, 1\}$ be the predicate for $x_j \notin S$. Then,

$$\Pr[E_S] = \mathbb{E}[\prod_{j=1}^{\ell} X_j].$$

Each X_j has expectation $\mathbb{E}[X_j] = 1 - s/n$ and variance $\mathrm{Var}(X_j) = (1 - s/n)s/n > s/2n$. By Fact 6.1,

$$\left| \Pr[E_S] - \left(1 - \frac{s}{n}\right)^{\ell} \right| \leq \left(c_1 \cdot \sqrt{\ell s / 2n} \right)^{c_2 t}.$$

Now, fix $I \subset [n]$ of size k. For any fixed value v, we have

$$\begin{aligned} |\Pr[y_I = v] - \Pr[w_I = v]| &\leq \sum_{S \subseteq I} \left| \Pr[y_S = 0^{|S|}] - \Pr[w_S = 0^{|S|}] \right| \\ &= \sum_{S \subseteq I} \cdot \left| \Pr[E_S] - (1 - s/n)^{\ell} \right| \\ &\leq 2^k \left(c_1 \cdot \sqrt{\ell k / 2n} \right)^{c_2 t}. \end{aligned}$$

Hence, the total variation distance between y_I and w_I is at most

$$d_{\mathsf{TV}}(y_I, w_I) \leq 2^{2k} \left(c_1 \cdot \sqrt{\ell k / 2n} \right)^{c_2 t}.$$

□

6.3 Attacks with Bounded Number of Samples

In the concrete estimates above, we have largely treated the number of samples available to the adversary and its running time as a single resource. For the statistical attacks considered in Section 6.2, the two complexity measures in fact coincide: The complexity of these attacks are all dominated by the number of independent query samples that are required to gain sufficient distinguishing advantage.

The situation is more subtle for algebraic attacks. In these attacks, additional running time can sometimes compensate for a smaller number of raw query samples by producing further samples that, while do not follow the original distribution, still satisfy the algebraic property exploited in the attack. This phenomenon already appears in the rank attack against LSN with mixture noise described in Section 6.1.2: the tensoring technique from [DKL09, CIMR26] allows for generating many more low-rank samples than are directly obtained from the PIR queries. Thus, unlike the statistical attacks above, algebraic attacks should not in general be viewed as purely sample-limited.

For our purposes, this distinction does not affect the parameter selection in an essential way, and we can still obtain significant gains in efficiency when aiming for security against bounded-sample adversaries. The algebraic attacks we identify are controlled by the explicit combinatorial barrier coming from the dummy coordinates, which can be made arbitrarily larger than the target running time by increasing L by a small additive amount. Since in our concrete regimes $\ell \approx q$ is already large, increasing $L - \ell$ by, say, 128 coordinates has negligible effect on the communication and online server work, while substantially increasing the cost of the best known algebraic attacks.

6.4 Comparison to Permuted Codes

Christ et al. [CGG$^+$26] recently introduced a cryptographic assumption called *permuted codes* which is related to the assumptions in this work.

The permuted codes assumption asserts the hardness of the permuted codes problem, where, just like in learning subspace with noise (LSN; Definition 6.1) the goal is to distinguish noisy random codewords sampled from a hidden code C from uniformly random samples. [CGG$^+$26] mostly considers the special case where C is a randomly permuted Reed-Solomon code, and the noise consists of Bernoulli i.i.d. (Hamming noise) and "alphabet permutation", namely random (global) re-labeling of the alphabet symbols.

One can view both permuted codes and PLDN in two different perspectives: as LSN problems [DKL09, CIMR26] or as permuted puzzles problems [BHW19].

In the first perspective, permuted codes and PLDN seem like two technically incomparable variants of LSN. There are several differences beyond the fact that they rely on different code structure.[7] First, in permuted codes the sampled codewords are uniformly random, while in PLDN the codewords always have low Hamming distance, potentially inducing more structure. Second, instead of an "alphabet permutation", in PLDN the only information that is given on the alphabet symbols is whether they are zero or non-zero. This is strictly stronger in the low-weight setting. Further, in the low-noise regime, permuted codes samples have a noticeable probability to be noiseless, while this is never the case in PLDN. This makes the alphabet permutation necessary in such regime to protect against algebraic attacks (similar the ones discussed above). In contrast, PLDN is plausibly secure with low noise even in the setting where the alphabet symbols are revealed (e.g. due to lack of entropy, see Section 6.1.2).

[7] While low-degree curves are essentially coordinate-wise concatenations of RS codewords, it is not clear how a reduction could map between the two distributions under the random permutation.

In the permuted puzzles framework, the "enhanced" toy conjecture from [BW21, BHMW21] may serve as a common basis for the two families: [CGG+26] shows it implies permuted codes, and [BHMW21] shows it implies sk-DEPIR by a simple adaptation of the schemes of [BIPW17, CHR17]. We can obtain low-storage sk-DEPIR based on analogous toy conjectures by similarly adapting our work: The lifted-RS construction (Theorem 5.2) can be based on "toy PLDN," where an entire punctured curve is masked by distractor noise á la [CHR17].

7 Performance

We implemented our two constructions in Go and C++ (for the online phase and encoding, resp.). The source code is available at `https://github.com/Caicai-Chen/FastSecretStatePIR`.

We benchmarked the performance of our implementations under selected choices of parameters, which we report in Tables 1 and 2. We consider parameter choices that correspond to a wide range of practical database sizes (from 0.8 GB to almost 100 TB) and provide at least 100 bits of conjectured security against both unbounded-sample adversaries and adversaries that are bounded to 2^{50} samples. Our security evaluations come from Conjectures 6.1 and 6.2, and are based on our cryptanalysis from Section 6. Considering lower (yet still reasonable) bounds on the adversary's sample complexity does not seem to give substantial gains in efficiency, as the best attack in the relevant regimes is the pairwise-intersections attack (Section 6.2.5), whose concrete complexity becomes significantly lower when t is further decreased.

In the benchmarked implementations, we set the parameters so that the polynomial interpolation coefficients, which the client uses at decoding, are trivial (i.e., all 1). This done by setting $\ell = q - 1$ in the lifted codes construction and $s = q^2 - 1$ in the construction based on concatenated decoding. We additionally set $L = \ell + 128$.

In Table 1, we additionally provide the performance of the original sk-PIR scheme based on standard Reed-Muller [BIPW17, CHR17] as a baseline comparison to our lifted-RS construction. All experiments were conducted on a machine equipped with an Apple Macbook M5 Pro (15-core CPU) with 48 GB of memory, running macOS 26.4. The reported (plaintext) database size is derived by dividing the encoding size by the respective code's rate: the RM construction has rate $\approx 1/m! \cdot t^m$ (Definition 3.5); for the lifted RS construction, we use the empirically measured rates, including those reported in Table 3; and for the concatenated RM construction, the rate is $\approx 1/m!$ (Theorem 4.1 and Corollary 4.2).

We additionally benchmark the performance of each of the steps separately as a function of the parameters in Figs. 7 and 8. The online phase consist of client-side query generation and server-side response computation, which simply consists of RAM retrieval. The server-side retrieval time for $q-1$ elements from a packed database of q^m entries is presented in Fig. 7. The results demonstrate that the server cost scales linearly in the number of retrieved entries. In particular, retrieving 2^{16} entries takes approximately 0.2 ms. The client-side query generation time is shown in Fig. 8. The measurements align with the theoretical $O(mtq)$ complexity: the runtime scales proportionally with q, and for fixed q, grows linearly with both the polynomial degree t and the number of polynomials m.

8 Acknowledgments

AI tools were used to assist with implementation, including scripts for parameter selection and Monte-Carlo rate estimation, concrete efficiency analysis of the Lifted RS construction, and estimating encoding runtimes. All AI-assisted analyses, estimates, code, and resulting claims were

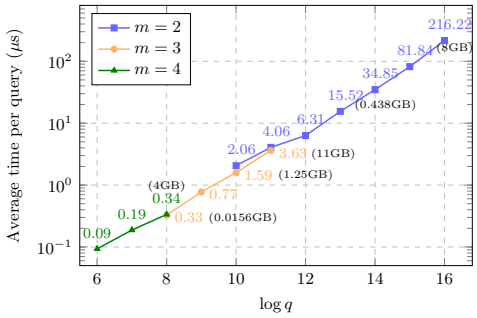

Figure 7: Benchmarks of the server answer by random access $q - 1$ entries from a database of q^m entries, where $q = 2^x$ with $6 \leq x \leq 16$. The database is stored in bit-packed form, representing each element of $\mathbb{F}_{2^x}$ using exactly x bits. The total database size is therefore $q^m \cdot x$ bits. The encoded database size (in GB) is indicated at selected points. Each data point corresponds to the average of over 1000 queries.

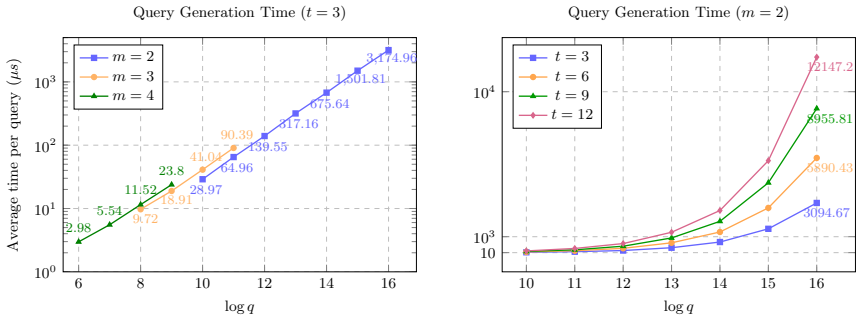

Figure 8: Benchmarks of the client query generation time. The left plot fixes $t = 3$ and varies $q = 2^x$ and m. The right plot fixes $m = 2$ and varies $q = 2^x$ and t. Each point corresponds to the average running time over 1000 independent queries.

independently reviewed and verified by the authors.

References

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

[AS26] Benny Applebaum and Shahar Shechter. Retrieve-compute PIR and its applications. Cryptology ePrint Archive, Paper 2026/1372, 2026.

[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. In *Proceedings of the 2025 ACM SIGSAC Conference on Computer and Communications Security, CCS 2025, Taipei, Taiwan, October 13-17, 2025*. ACM, 2025.

[BF90] Donald Beaver and Joan Feigenbaum. Hiding instances in multioracle queries. In *Lecture Notes in Computer Science*, pages 37–48. Springer Berlin Heidelberg, 1990.

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

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

[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.

[BIP+18] Dan Boneh, Yuval Ishai, Alain Passelègue, Amit Sahai, and David J. Wu. Exploring crypto dark matter: - new simple PRF candidates and their applications. In Amos Beimel and Stefan Dziembowski, editors, *Theory of Cryptography - 16th International Conference, TCC 2018, Panaji, India, November 11-14, 2018, Proceedings, Part II*, volume 11240 of *Lecture Notes in Computer Science*, pages 699–729. Springer, 2018.

[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.

[BJS07] Alin Bostan, Claude-Pierre Jeannerod, and Éric Schost. Solving toeplitz- and vandermonde-like linear systems with large displacement rank. In *Proceedings of ISSAC 2007*, pages 33–40, 2007.

[BS04] Alin Bostan and Éric Schost. On the complexities of multipoint evaluation and interpolation, 2004. Preprint.

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

[CGG+13] Alain Couvreur, Philippe Gaborit, Valérie Gauthier-Umaña, Ayoub Otmani, and Jean-Pierre Tillich. Distinguisher-based attacks on public-key cryptosystems using reed-solomon codes. *CoRR*, abs/1307.6458, 2013.

[CGG+26] Miranda Christ, Noah Golowich, Sam Gunn, Ankur Moitra, and Daniel Wichs. Improved pseudorandom codes from permuted puzzles. In *Proceedings of STOC 2026*, 2026. Full version: Cryptology ePrint Archive, Paper 2025/2222.

[CGKO06] Reza Curtmola, Juan Garay, Seny Kamara, and Rafail Ostrovsky. Searchable symmetric encryption: improved definitions and efficient constructions. In *Proceedings of the 13th ACM Conference on Computer and Communications Security (CCS)*, pages 79–88. ACM, 2006.

[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.

[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.

[CIMR26] Caicai Chen, Yuval Ishai, Tamer Mour, and Alon Rosen. Secret-key PIR from random linear codes. In *STOC 2026*, 2026. Full version: `https://eprint.iacr.org/2025/646`.

[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.

[CXY20] Ronald Cramer, Chaoping Xing, and Chen Yuan. Efficient multi-point local decoding of reed-muller codes via interleaved codex. *IEEE Trans. Inf. Theory*, 66(1):263–272, 2020.

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

[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.

[GK14] Alan Guo and Swastik Kopparty. List-decoding algorithms for lifted codes. *arXiv preprint arXiv:1412.0305*, 2014.

[GK16] Alan Guo and Swastik Kopparty. List-decoding algorithms for lifted codes. *IEEE Transactions on Information Theory*, 62(5):2719–2725, 2016.

[GKS13] Alan Guo, Swastik Kopparty, and Madhu Sudan. New affine-invariant codes from lifting. In *Proceedings of ITCS 2013*, pages 529–540, 2013.

[GLM+24] Ashrujit Ghoshal, Baitian Li, Yaohua Ma, Chenxin Dai, and Elaine Shi. Information-theoretic multi-server PIR with global preprocessing. *IACR Cryptol. ePrint Arch.*, page 765, 2024.

[GLR+91] Peter Gemmell, Richard Lipton, Ronitt Rubinfeld, Madhu Sudan, and Avi Wigderson. Self-testing/correcting for polynomials and for approximate functions. In *Proceedings of the Twenty-Third Annual ACM Symposium on Theory of Computing*, STOC '91, page 33–42, New York, NY, USA, 1991. Association for Computing Machinery.

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

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

[GS92] Peter Gemmell and Madhu Sudan. Highly resilient correctors for polynomials. *Inf. Process. Lett.*, 43(4):169–174, September 1992.

[GY20] Parikshit Gopalan and Amir Yehudayoff. Concentration for limited independence via inequalities for the elementary symmetric polynomials. *Theory of Computing*, 16(17):1–29, 2020.

[HMMM14] Sardar Anisul Haque, Farnam Mansouri, and Marc Moreno Maza. On the parallelization of subproduct tree techniques targeting many-core architectures. In Vladimir P. Gerdt, Wolfram Koepf, Werner M. Seiler, and Evgenii V. Vorozhtsov, editors, *Computer Algebra in Scientific Computing*, pages 171–185, Cham, 2014. Springer International Publishing.

[HOW15] Brett Hemenway, Rafail Ostrovsky, and Mary Wootters. Local correctability of expander codes. *Inf. Comput.*, 243:178–190, 2015.

[HPPV20] Lukas Holzbaur, Rina Polyanskaya, Nikita Polyanskii, and Ilya Vorobyev. Lifted reed-solomon codes with application to batch codes. *arXiv preprint arXiv:2001.11981*, 2020.

[HR26] Alexandra Henzinger and Seyoon Ragavan. Two-server private information retrieval in sublinear time and quasilinear space. In *Advances in Cryptology – EUROCRYPT 2026*, pages 64–94, 2026.

[HRS13] Elad Haramaty, Noga Ron-Zewi, and Madhu Sudan. Absolutely sound testing of lifted codes. In Prasad Raghavendra, Sofya Raskhodnikova, Klaus Jansen, and José D. P. Rolim, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques - 16th International Workshop, APPROX 2013, and 17th International Workshop, RANDOM 2013, Berkeley, CA, USA, August 21-23, 2013. Proceedings*, volume 8096 of *Lecture Notes in Computer Science*, pages 671–682. Springer, 2013.

[IKOS04] Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, and Amit Sahai. Batch Codes and Their Applications. In *Proceedings of the 36th Annual ACM Symposium on Theory of Computing*, STOC '04, pages 262–271, New York, NY, USA, 2004. Association for Computing Machinery. Full version: `https://web.cs.ucla.edu/~rafail/PUBLIC/62.pdf`.

[KC21] Dmitry Kogan and Henry Corrigan-Gibbs. Private blocklist lookups with checklist. In *30th USENIX Security Symposium (USENIX Security 21)*, pages 875–892, 2021.

[KO97] E. Kushilevitz and R. Ostrovsky. Replication is not needed: single database, computationally-private information retrieval. In *Proceedings 38th Annual Symposium on Foundations of Computer Science*, pages 364–373, 1997.

[KS99] Nicholas M Katz and Peter Sarnak. *Random matrices, Frobenius eigenvalues, and monodromy*, volume 45. American Mathematical Soc., 1999.

[KSY11] Swastik Kopparty, Shubhangi Saraf, and Sergey Yekhanin. High-rate codes with sublinear-time decoding. In Lance Fortnow and Salil P. Vadhan, editors, *Proceedings of the 43rd ACM Symposium on Theory of Computing, STOC 2011, San Jose, CA, USA, 6-8 June 2011*, pages 167–176. ACM, 2011.

[LHP+21] Hedongliang Liu, Lukas Holzbaur, Nikita Polyanskii, Sven Puchinger, and Antonia Wachter-Zeh. Quadratic-curve-lifted reed–solomon codes, 2021. arXiv preprint arXiv:2109.14478.

[LLFP24] Arthur Lazzaretti, Zeyu Liu, Ben Fisch, and Charalampos Papamanthou. Multi-server doubly efficient PIR. Cryptology ePrint Archive, Paper 2024/829, 2024. `https://eprint.iacr.org/2024/829`.

[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.

[LN19] Julien Lavauzelle and Jade Nardi. Weighted lifted codes: Local correctabilities and application to robust private information retrieval, 2019. arXiv preprint arXiv:1904.08696.

[LN21] Julien Lavauzelle and Jade Nardi. Weighted lifted codes: Local correctabilities and application to robust private information retrieval. *IEEE Transactions on Information Theory*, 67(1):111–123, 2021.

[LP24] Arthur Lazzaretti and Charalampos Papamanthou. Single pass client-preprocessing private information retrieval. In Davide Balzarotti and Wenyuan Xu, editors, *33rd USENIX Security Symposium, USENIX Security 2024, Philadelphia, PA, USA, August 14-16, 2024*. USENIX Association, 2024.

[Man14] Farnam Mansouri. On the parallelization of integer polynomial multiplication. 2014.

[MB72] Robert T. Moenck and Allan Borodin. Fast modular transforms via division. In *Scandinavian Workshop on Algorithm Theory*, 1972.

[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`.

[Mul54] David E. Muller. Application of boolean algebra to switching circuit design and to error detection. *Trans. I R E Prof. Group Electron. Comput.*, 3:6–12, 1954.

[NVI10] NVIDIA Corporation. Tesla M2050/M2070 GPU Computing Module. Datasheet, NVIDIA Corporation, April 2010.

[NVI26] NVIDIA Corporation. NVIDIA H100 Tensor Core GPU. `https://www.nvidia.com/en-us/data-center/h100/`, 2026. Product specifications; accessed 2026-07-17.

[OPPW23] Hiroki Okada, Rachel Player, Simon Pohmann, and Christian Weinert. Towards practical doubly-efficient private information retrieval. *IACR Cryptol. ePrint Arch.*, page 1510, 2023.

[PPY18] Sarvar Patel, Giuseppe Persiano, and Kevin Yeo. Private stateful information retrieval. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pages 1002–1019, 2018.

[PY22] Giuseppe Persiano and Kevin Yeo. Limits of preprocessing for single-server PIR. In Joseph (Seffi) Naor and Niv Buchbinder, editors, *Proceedings of the 2022 ACM-SIAM Symposium on Discrete Algorithms, SODA 2022, Virtual Conference / Alexandria, VA, USA, January 9 - 12, 2022*, pages 2522–2548. SIAM, 2022.

[Ree54] I. Reed. A class of multiple-error-correcting codes and the decoding scheme. *Transactions of the IRE Professional Group on Information Theory*, 4(4):38–49, 1954.

[RS60] I. S. Reed and G. Solomon. Polynomial codes over certain finite fields. *Journal of the Society for Industrial and Applied Mathematics*, 8(2):300–304, 1960.

[Ser16] Jean-Pierre Serre. *Topics in Galois theory*. CRC Press, 2016.

[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.

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

[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.

[Wie06] Christian Wieschebrink. Two np-complete problems in coding theory with an application in code based cryptography. In *Proceedings of the 2006 IEEE International Symposium on Information Theory (ISIT)*, pages 1733–1737. IEEE, July 2006.

[XZO+26] Pan Xiao, Heng Zhang, Rending Ouyang, Cong Zhang, Jian Liu, Kui Ren, and Chun Chen. Towards making doubly-efficient PIR practical. Cryptology ePrint Archive, Paper 2026/243, 2026.

[Yan25] Tal Yankovitz. Asymptotically good large-alphabet ldcs with polylogarithmic query complexity. *Electron. Colloquium Comput. Complex.*, TR25, 2025.

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

A Proof of Claim 6.2

For $S \subseteq \Omega$ with $|S| \leq t$, define

$$Q_X(S) = \sum_{\substack{A \supseteq S \\ |A| \leq t}} (-1)^{|A|-|S|} \Pr[A \subseteq X],$$

and define $Q_Y(S)$ similarly. By assumption, $Q_X(S) = Q_Y(S)$ for every $|S| \leq t$.

For a fixed realization $B \subseteq \Omega$, the corresponding truncated expression is

$$\mathbf{1}[S \subseteq B] \sum_{j=0}^{t-|S|} (-1)^j \binom{|B \setminus S|}{j}.$$

If $B = S$, this expression equals 1. If $S \subsetneq B$ and $|B| \leq t$, it equals 0 by the binomial identity. Thus it agrees with $\mathbf{1}[B = S]$ whenever $|B| \leq t$. If $S \subseteq B$ and $|B| > t$, the absolute error is at most

$$\binom{|B \setminus S|}{t+1-|S|}.$$

Therefore

$$|\Pr[X = S] - Q_X(S)| \leq \sum_{\substack{A \supseteq S \\ |A|=t+1}} \Pr[A \subseteq X],$$

and the same bound holds for Y.

Summing over all S with $|S| \leq t$, each fixed $(t+1)$-set A contains at most 2^{t+1} such subsets S. Hence

$$\sum_{|S| \leq t} |\Pr[X = S] - Q_X(S)| \leq 2^{t+1} \mathbb{E}\binom{|X|}{t+1},$$

and similarly for Y. For the tail $|S| > t$, we use

$$\Pr[|X| > t] \leq \mathbb{E}\binom{|X|}{t+1}, \qquad \Pr[|Y| > t] \leq \mathbb{E}\binom{|Y|}{t+1}.$$

Since $Q_X(S) = Q_Y(S)$ for all $|S| \leq t$, combining these bounds gives

$$\sum_{S \subseteq \Omega} |\Pr[X = S] - \Pr[Y = S]| \leq 2^{t+1} \left(\mathbb{E}\binom{|X|}{t+1} + \mathbb{E}\binom{|Y|}{t+1} \right).$$

Dividing by 2 proves the claim.

Parameters	RM + standard decoding	Lifted RS
Security with Unbounded Number of Samples		
$q = 2^{16}$, $m = 2$, $t = 8$ Storage: 8.589 GB Security: 112 bits	DB: 0.0671 GB (Rate: 0.008) Probes: 1.96×10^{-3} ($\uparrow$ 0.263 MB, $\downarrow$ 0.131 MB) Client Runtime: $8.9 + 2.5$ ms	DB: 0.7800 GB (Rate: 0.09) Probes: 1.68×10^{-4} ($\uparrow$ 0.263 MB, $\downarrow$ 0.131 MB) Client Runtime: $8.9 + 2.5$ ms
$q = 2^{17}$, $m = 2$, $t = 7$ Storage: 36.507 GB Security: 102 bits	DB: 0.3725 GB (Rate: 0.01) Probes: 7.48×10^{-4} ($\uparrow$ 0.558 MB, $\downarrow$ 0.279 MB) Client Runtime: $19.04 + 6.3$ ms	DB: 4.342 GB (Rate: 0.12) Probes: 6.42×10^{-5} ($\uparrow$ 0.558 MB, $\downarrow$ 0.279 MB) Client Runtime: $19.04 + 6.3$ ms
$q = 2^{18}$, $m = 2$, $t = 7$ Storage: 154.619 GB Security: 108 bits	DB: 1.578 GB (Rate: 0.01) Probes: 3.74×10^{-4} ($\uparrow$ 1.180 MB, $\downarrow$ 0.590 MB) Client Runtime: $41.25 + 15.6$ ms	DB: 20.04 GB (Rate: 0.13) Probes: 2.94×10^{-5} ($\uparrow$ 1.180 MB, $\downarrow$ 0.590 MB) Client Runtime: $41.25 + 15.6$ ms
Security with Bounded Number of Samples (2^{50} Samples)		
$q = 2^{12}$, $m = 2$, $t = 6$ Storage: 25 MB Security: 289 bits	DB: 0.35 MB (Rate: 0.014) Probes: 1.76×10^{-2} ($\uparrow$ 0.012 MB, $\downarrow$ 0.006 MB) Client Runtime: $0.8 + 0.16$ ms	DB: 2.95 MB (Rate: 0.117) Probes: 2.08×10^{-3} ($\uparrow$ 0.012 MB, $\downarrow$ 0.006 MB) Client Runtime: $0.8 + 0.16$ ms
$q = 2^{15}$, $m = 2$, $t = 5$ Storage: 2.013 GB Security: 450 bits	DB: 0.04 GB (Rate: 0.02) Probes: 1.53×10^{-3} ($\uparrow$ 0.123 MB, $\downarrow$ 0.061 MB) Client Runtime: $4.2 + 1.3$ ms	DB: 0.41 GB (Rate: 0.2) Probes: 1.51×10^{-4} ($\uparrow$ 0.123 MB, $\downarrow$ 0.061 MB) Client Runtime: $4.2 + 1.3$ ms
$q = 2^{18}$, $m = 2$, $t = 5$ Storage: 154.6 GB Security: 648 bits	DB: 3.1 GB (Rate: 0.02) Probes: 1.9×10^{-4} ($\uparrow$ 1.18 MB, $\downarrow$ 0.59 MB) Client Runtime: $32.8 + 22.7$ ms	DB: 36.6 GB (Rate: 0.237) Probes: 1.61×10^{-5} ($\uparrow$ 1.18 MB, $\downarrow$ 0.59 MB) Client Runtime: $32.8 + 22.7$ ms
$q = 2^{20}$, $m = 2$, $t = 5$ Storage: 2.75 TB Security: 800 bits	DB: 55 GB (Rate: 0.02) Probes: 4.77×10^{-5} ($\uparrow$ 5.243 MB, $\downarrow$ 2.621 MB) Client Runtime: $127.7 + 197.2$ ms	DB: 712 GB (Rate: 0.259) Probes: 3.68×10^{-6} ($\uparrow$ 5.243 MB, $\downarrow$ 2.621 MB) Client Runtime: $127.7 + 197.2$ ms

Table 1: Concrete efficiency of our lifted-RS construction (Theorem 5.2) under selected parameter choices (q, m, t), with the RM-based construction [BIPW17, CHR17] as a baseline for comparison. We consider two adversarial settings: adversaries that can observe an unbounded number of queries and adversaries that are limited to 2^{50} samples. The first column specifies the parameters, which determine the server-side storage, i.e. size of DB encoding $q^m \log(q)$, and the estimated security level in bits based on Conjecture 6.1, which we evaluate at $\log_2(T)$ bits if the protocol is concretely $(T, *)$-secure and, resp., if the protocol is concretely $(T, 2^{50})$-secure. The remaining two columns provide, for each construction, the database size before encoding, the probe complexity (measured as the ratio between the number of RAM accesses per query and the number of entries in the original database), the upload/download communication per query and the client runtime of the implementation on an Apple Macbook M5 Pro (15-core CPU) with 48 GB of memory, running macOS 26.4.

Parameters	**RM + conc. decoding**	**Parameters**	**RM + conc. decoding**
Security with Unbounded Number of Samples		Security with Bounded Number of Samples (2^{50} Samples)	
$q = 2^{12}$, $m = 3$, $t = 6$ Storage: 103.08 GB Security: 114.5	DB: 17.18 GB (Rate: 0.167) Probes: 8.78×10^{-3} ($\uparrow$ 0.453 GB, $\downarrow$ 0.151 GB) Client Runtime: 0.188 s + 117.1 ms	$q = 2^{12}$, $m = 3$, $t = 4$ Storage: 103.08 GB Security: 936 bits	DB: 17.18 GB (Rate: 0.167) Probes: 5.86×10^{-3} ($\uparrow$ 0.302 GB, $\downarrow$ 0.101 GB) Client Runtime: 0.125 s + 77.9 ms
$q = 2^{13}$, $m = 3$, $t = 6$ Storage: 893.35 GB Security: 116.5	DB: 148.9 GB (Rate: 0.167) Probes: 4.4×10^{-3} ($\uparrow$ 1.963 GB, $\downarrow$ 0.654 GB) Client Runtime: 0.721 s + 476.9 ms	$q = 2^{13}$, $m = 3$, $t = 4$ Storage: 893.353 GB Security: $>$ 1000 bits	DB: 148.9 GB (Rate: 0.167) Probes: 2.93×10^{-3} ($\uparrow$ 1.309 GB, $\downarrow$ 0.436 GB) Client Runtime: 0.489 s + 317.6 ms
$q = 2^{14}$, $m = 3$, $t = 6$ Storage: 7.696 TB Security: 118.5	DB: 1.283 TB (Rate: 0.167) Probes: 2.20×10^{-3} ($\uparrow$ 8.455 GB, $\downarrow$ 2.818 GB) Client Runtime: 2.75 s + 2.231 s	$q = 2^{14}$, $m = 3$, $t = 4$ Storage: 7.696 TB Security: $>$ 1000 bits	DB: 1.283 TB (Rate: 0.167) Probes: 1.465×10^{-3} ($\uparrow$ 5.637 GB, $\downarrow$ 1.879 GB) Client Runtime: 1.867 s + 1.482 s
$q = 2^{16}$, $m = 3$, $t = 6$ Storage: 562.95 TB Security: 122.5	DB: 93.82 TB (Rate: 0.167) Probes: 5.49×10^{-4} ($\uparrow$ 154.62 GB, $\downarrow$ 51.54 GB) Client Runtime: 45.2 s + 46.85 s	$q = 2^{16}$, $m = 3$, $t = 4$ Storage: 562.95 TB Security: $>$ 1000 bits	DB: 93.82 TB (Rate: 0.167) Probes: 3.66×10^{-4} ($\uparrow$ 103.08 GB, $\downarrow$ 34.36 GB) Client Runtime: 30.6 s + 31.05 s

Table 2: Concrete efficiency of our construction based on RM with concatenated-curve decoding (Theorem 5.3), under selected parameter choices (q, m, t). We consider two adversarial settings: adversaries that can observe an unbounded number of queries and adversaries that are limited to 2^{50} samples. The first column specifies the parameters, which determine the server-side storage, i.e. size of DB encoding $q^m \log(q)$, and the estimated security level in bits based on Conjecture 6.2, which we evaluate at $\log_2(T)$ bits if the protocol is concretely $(T, *)$-secure and, resp., if the protocol is concretely $(T, 2^{50})$-secure. The second column provides the database size before encoding, the probe complexity (measured as the ratio between the number of RAM accesses per query and the number of entries in the original database), the upload/download communication per query and the client runtime of the implementation on an Apple Macbook M5 Pro (15-core CPU) with 48 GB of memory, running macOS 26.4.

m	q	t	rate($\mathsf{Lift\text{-}RS}_q(m,t)$)	rate($\mathsf{RM}_q(m,t)$) $a \approx 1/(t^m m!)$	ratio
2	2^{12}	3	0.3684	0.05556	6.63
		4	0.2482	0.03125	7.94
		5	0.1598	0.02	7.99
		7	0.0848	0.0102	8.31
		8	0.0673	0.007812	8.61
		10	0.0444	0.005	8.87
	2^{15}	3	0.4515	0.05556	8.13
		4	0.31	0.03125	9.92
		5	0.2024	0.02	10.12
		7	0.1058	0.0102	10.37
		8	0.0864	0.007812	11.06
		10	0.0542	0.005	10.84
	2^{18}	3	0.5059	0.05556	9.11
		4	0.3541	0.03125	11.33
		5	0.2367	0.02	11.83
		7	0.1283	0.0102	12.57
		8	0.1021	0.007812	13.07
		10	0.0679	0.005	13.58
3	2^8	3	0.0404	0.006173	6.55
		4	0.0184	0.002604	7.05
		5	0.0089	0.001333	6.71
		7	0.0032	0.000486	6.69
		8	0.0020	0.000326	6.14
		10	0.0013	0.000167	7.5
	2^{10}	3	0.0512	0.006173	8.3
		4	0.0241	0.002604	9.25
		5	0.0112	0.001333	8.36
		7	0.0042	0.000486	8.54
		8	0.0029	0.000326	8.76
		10	0.0014	0.000167	8.7
	2^{12}	3	0.0611	0.006173	9.9
		4	0.0277	0.002604	10.62
		5	0.0141	0.001333	10.57
		7	0.0047	0.000486	9.67
		8	0.0037	0.000326	11.52
		10	0.0018	0.000167	10.8
4	2^6	3	0.0042	0.000514	8.07
		4	0.0013	0.000163	7.68
		5	0.00060	0.000067	9
		7	0.00010	0.000017	5.76
		8	0.00010	0.00001	9.83
		10	0.00005	0.000004	12
	2^8	3	0.0045	0.000514	8.75
		4	0.0012	0.000163	7.37
		5	0.00050	0.000067	7.5
		7	0.00005	0.000017	2.88
		8	0.00005	0.00001	4.92
	2^{10}	3	0.0041	0.000514	7.97
		4	0.0015	0.000163	9.22
		5	0.00075	0.000067	11.25

Table 3: Example empirical rate estimates for the curve-lifted code $\mathsf{Lift\text{-}RS}_q(m,t)$ over characteristic two. The RM comparator is $\approx 1/(t^m m!)$ corresponding to an m-variate Reed–Muller code of total degree on the order of q/t when q is large and m is constant. The last column represents the ratio between the rate of the two codes.