

A Survey of Locally Decodable Codes and Private Information Retrieval Schemes

Eric Hielscher

July 10, 2007

Abstract

We survey the state of the art in the fields of Locally Decodable Codes and Private Information Retrieval. We present enough background, including a brief description of Quantum Computing, to make this paper self-contained. Of note, we include the recent construction of Yekhanin [29, 14], which yields the best known general bounds for PIR protocols and LDCs.

1 Introduction

Coding theory, which provides much of the setting and foundation for the topics of this paper, can be roughly divided into two separate but intimately related fields: the study of error-correcting codes, and information theory [25]. The study of error-correcting codes goes back to the beginnings of information theory with Shannon's and Hamming's seminal papers [11, 23, 24], published nearly simultaneously in the late 1940's, and these codes have since found numerous applications in real life situations where noise is an important consideration.

An error-correcting code is a mapping of strings x of length n to codewords $C(x)$ of length m , with m typically much greater than n ; we then also define some decoding algorithm which attempts to recover strings x from their codewords $C(x)$. The information rate (or just rate) of a code is thus the ratio n/m , which can be interpreted as the average amount of information from the original message x per symbol of the codeword $C(x)$. The increase in length allows one to introduce some redundancy into the codeword space, such that if some portion of the codeword is corrupted the original data is not lost. This is done by using a sparse space of codewords so that any two codewords have a sufficient number of positions different from each other, and thus if the number of corrupted positions in a codeword is low the decoder is able to determine the correct uncorrupted codeword $C(x)$ and thus to decode the original message x . This notion of distance was first formalized by Hamming and has come to be known as the *Hamming Distance* between two sequences of elements from a finite space; we write $\Delta(x, y)$ to denote the Hamming distance between two sequences x and y . Other important parameters in the design of error-correcting codes include the minimum distance $\Delta(C)$ of a code C , defined as $\min_{x \neq y \in C} \Delta(x, y)$, i.e. the minimum Hamming distance between any pair of codewords in C ; and the relative distance of a code, defined as the ratio $\Delta(C)/m$ between the code's minimum distance and the length of the codewords. In Hamming's original paper he defined the first error-correcting codes, which have since come to be known as Hamming codes, and these codes are part of a class of codes called perfect codes due to the fact they have maximum rate with respect to their minimum distance. Numerous other codes have been designed since, including for example Reed-Solomon codes which are used today to encode the information on compact discs so as to protect against the noise introduced by the decoding process and by scratches

on the surface of the medium - in fact without the use of such codes, compact discs would be unusable due to excess noise from small scratches.

A field intimately related to that of error-correcting codes is information theory, whose history goes back to the immensely influential publication by Claude Shannon [23]. In his work, Shannon developed a mathematical formalization of the concept of information through his definition of entropy, or the amount of information which one is missing when one doesn't know the value of some random variable. Using this he went on to study how one can efficiently communicate information across both noiseless and noisy channels. In the former class of channels the major concern is data compression, i.e. trying to transmit as few symbols across the channel as possible such that the party on the other end is still able to recover the intended message. In the latter class, there is the added issue that the channel probabilistically introduces noise into the message, resulting in symbols being received which, while within the alphabet of the code, were not those intended to be sent. Though the study of noisy channels is more directly relevant to our present topic, Shannon showed that the same notion of information rate is applicable to both classes of channels.

For a useful survey on the broader aspects of coding theory, we direct the reader to [25].

1.1 Locally Decodable Codes

Locally Decodable Codes (LDCs) are a rather recent addition to coding theory. The first paper to explicitly cover the subject was by Babai *et al.* [1] in 1991, which examined it from the perspective of computational verification and holographic proofs. The power which LDCs have over traditional error-correcting codes is that in the event that one is only interested in decoding a small number of bits of a message x from a codeword $C(x)$ one need only inspect a small number of bits of $C(x)$, and thus the decoding algorithm can be faster than one which would have to inspect all or most of $C(x)$. A real-world example in which this could be useful is the decompression of an encoded archive file. If one is only interested in one file out of thousands in a large archive, the decoding algorithm can run much faster if it need not decode the entire archive but only a small part of it. The study of Locally Decodable Codes has also proven useful in various areas of theoretical computer science, including worst-case to average-case reductions [2, 26], extractors [19], probabilistically checkable proofs [1], and self-correcting computation [8].

1.2 Private Information Retrieval

Private Information Retrieval, first introduced by Chor, Goldreich, Kushilevitz, and Sudan [5], deals with the situation where a user wants to access data from some database while preventing the server(s) which house the database from gaining any information about which pieces of data are being retrieved. One example in which such a protocol would be useful is in the stock-trading domain. It might be the case that a trader wishes to query the current price of a given stock, but also wishes to keep the identity of the stock secret to avoid attracting attention to the stock. We note that PIR is largely a cryptographic study, with the scarce resource being *communication*. In other words, computation is assumed to be cheap, while transmission of information between parties is relatively expensive. A trivial protocol would be for the user to simply request the entire database from a server, requiring n bits of communication. In the case where there is only one server, unfortunately, full information-theoretic privacy requires $\Omega(n)$ communication and thus we can come up with no scheme better than this trivial protocol [5]. Thus Chor *et al.* proposed the idea of redundantly storing the database in k different servers, which in the basic case are assumed not to communicate with each other. In this setting one can devise protocols which guarantee the information-theoretic privacy of the index i while also

using far less communication. The initial upper bound on the communication complexity of PIR for a generic k number of servers was $O(n^{1/k})$ given in [5], and has since been improved to $n^{O(\log \log k / (k \log k))}$. This latter bound was first given in [3], with the precise bound being $O(n^{2 \log \log k / (k \log k)})$, but the constant in the exponent can be improved by combining a result of Yekhanin [29, 14] with the generic recursion technique given in *et al.* [3].

Many variations of this model have been studied. One example is Computational PIR [18], which trades information-theoretic privacy for computationally bounded privacy, i.e. making it computationally expensive (reminiscent of current cryptographic schemes) for the servers to gain any information about the index i . In this case much better upper bounds are possible, with schemes now known that achieve polylogarithmic complexity in n . Another related model is that of (information-theoretic) Quantum PIR, in which the communication is done using quantum bits (qubits) rather than classical bits and the parties use quantum computers. The results of [15] use quantum tools to prove both an exponential lower bound for 2-query LDCs as well as showing a technique to reduce two classical queries in a PIR scheme or LDC decoder to one quantum query, thus giving evidence (though not proof) of a quantum/classical separation in this domain. Another model is that of Symmetric PIR (SPIR), in which the user is required *only* to learn about the bits in which she is interested and nothing else. Gertner *et al.* proved in [9] that SPIR is impossible unless the servers are allowed to share a random string, the generation of which is either expensive in terms of communication (i.e. a synchronization among the servers of some random string prior to each user transaction) or in terms of memory consumption (i.e. the storage of many pre-calculated random strings to be used over the course of many transactions). SPIR has even been studied in the quantum setting as QSPIR [16]. Yet other variations allow for collusions of up to t servers while still guaranteeing privacy (t -private PIR), etc.

There is a natural connection between PIR schemes and LDCs. Intuitively one can think of each query to an LDC codeword as a query to a server in a PIR scheme, with the length w of each server's response and the size of the alphabet Σ of the LDC being related by the following formula:

$$|\Sigma| = 2^w.$$

Thus there is a one-to-one mapping between the symbols in Σ and the possible responses that a server could return. Precise details of such a translation will be given in Section 4.1.

2 Preliminaries

Throughout this paper we use the following notation:

$$\begin{aligned} [n] &:= \{1, \dots, n\}. \\ Y_\ell &:= \text{the } \ell\text{th bit of the string } Y. \end{aligned}$$

We also use various dot products throughout this paper. We use the following conventions with respect to these dot products:

$$\begin{aligned} a \cdot b &:= \sum_{i=1}^{|a|} a_i b_i \pmod{2}. \\ \langle a, b \rangle &:= \sum_{i=1}^{|a|} a_i * b_i, \text{ where } * \text{ is the multiplicative operation of the field.} \end{aligned}$$

Thus the dot is used for base 2 dot products, and the angle brackets are used for general dot products with the field given by the context.

Finally, all logarithms are base 2 unless otherwise noted.

2.1 Information Theory

In this section we first give a brief overview of the key concepts from Information Theory, as these both form the basis for much of coding theory as well prove useful in the analysis of LDCs and PIR schemes. For a complete reference to the subject we refer the reader to [7].

Definition 2.1. The **Hamming Distance** between two strings x and y of equal length, which we denote by $\Delta(x, y)$, is the number of positions on which x and y differ.

One can also view the strings x and y as n -dimensional vectors in the space \mathbb{R}^n . In this case the strings form the vertices of an n -dimensional hypercube, and the Hamming Distance between any two strings is equal to the *Manhattan Distance* between their respective vertices (i.e. the least number of edges traversed on any walk between them).

Definition 2.2. The **Shannon Entropy** (or just **Entropy**) $\mathcal{H}(X)$ of a discrete random variable $X \in \{x_1, \dots, x_n\}$ is defined as

$$\mathcal{H}(X) = \sum_{i=1}^n p(x_i) \log(1/p(x_i)) = - \sum_{i=1}^n p(x_i) \log(p(x_i)),$$

where $p(x_i) = \Pr[X = x_i]$ is the probability mass function of the random variable X .

Shannon Entropy is a measure of the smallest possible number of bits needed to transmit a given message across a channel. Alternatively, one can view it as a measure of the amount of information which the recipient of a message lacks until she receives it. In the context of coding theory, we apply the concept of Entropy by treating each bit as a discrete random variable which ranges over its alphabet. Shannon Entropy, as it represents a theoretical lower limit on string length, is often used in proving lower bounds and is employed in this capacity in some of the lower bound proofs in Section 5. One useful special case of Entropy is called the Joint Entropy $\mathcal{H}(X, Y)$ of two discrete random variables X and Y , and is defined as

$$\mathcal{H}(X, Y) = - \sum_{x, y \in X, Y} p(x, y) \log(p(x, y)).$$

The Joint Entropy is, as its name implies, a measure of the entropy of two discrete random variables treated as one system. Another useful extension of the concept of Entropy is as that of Conditional Entropy.

Definition 2.3. The **Conditional Entropy** $\mathcal{H}(X|Y)$ of two discrete random variables X and Y is defined as

$$\mathcal{H}(X|Y) = \mathcal{H}(X, Y) - \mathcal{H}(Y).$$

One can think of the Conditional Entropy $\mathcal{H}(X|Y)$ as measuring the amount of information content left in the variable X , whose value one doesn't yet know, when one already knows the value of the variable Y . In other words, it measures how much learning the value of Y tells you about the variable X .

Definition 2.4. The **Mutual Information** $I(X; Y)$ between two discrete random variables X and Y is defined as

$$I(X; Y) = \sum_{x \in X} \sum_{y \in Y} p(x, y) \frac{p(x, y)}{p(x)p(y)},$$

or equivalently

$$I(X; Y) = \mathcal{H}(X) - \mathcal{H}(X|Y),$$

where $p(x, y)$ is the joint probability distribution of X and Y , and $p(x)$ and $p(y)$ are the marginal probability distributions of X and Y , respectively.

Mutual Information is a measure of the amount of mutual dependence between two random variables.

2.2 Locally Decodable Codes

First, we define a code C to be a function which maps strings of symbols from an input alphabet to strings of symbols from an output alphabet. For alphabets Γ and Σ , we write $C : \Gamma^n \rightarrow \Sigma^m$ to denote a code C which maps strings of length n from Γ to strings of length m from Σ .

Next, we formally define a locally decodable code.

Definition 2.5. A (q, δ, ϵ) -**Locally Decodable Code (LDC)** is a code $C : \Gamma^n \rightarrow \Sigma^m$ such that for any message $x \in \Gamma^n$ and any string $y \in \Sigma^m$ for which $\Delta(C(x), y) \leq \delta m$, any character x_i can be recovered with probability $\geq 1/|\Gamma| + \epsilon$ by a probabilistic algorithm which makes at most q non-adaptive queries to characters of y . Such a decoding algorithm is called a (q, δ, ϵ) -local decoding algorithm for C .

One usually thinks of the parameters δ and ϵ as constants. The main parameters of interest in LDCs are the query complexity q and the code length m , both of which we would like to minimize. Since these two parameters are negatively correlated, however, the problem becomes an analysis of the tradeoff between lower query complexity and lower code length. While in the definition we focus solely on non-adaptive queries, it has been shown in [13] that one can simulate any adaptive locally decoding algorithm with a non-adaptive algorithm with only a constant loss in error probability.

In all known LDC constructions, it is possible to partition the indices of the codeword into tuples of at most size q such that the decoding algorithm reads each tuple in this partition with equal probability. Indeed, this fact should be somewhat natural, as an adversary who knew that some particular indices were being read more frequently than others could strategically corrupt these indices and reduce the performance of the decoding algorithm. This observation leads one to the notion of a *smooth code*.

Definition 2.6. A (q, c, ϵ) -**Smooth Code** is a code $C : \Gamma^n \rightarrow \Sigma^m$ such that for any message x , any character x_i can be recovered with probability $\geq 1/|\Gamma| + \epsilon$ by a probabilistic algorithm which makes at most q queries to indices of $C(x)$. Further, the algorithm queries any given index $C(x)_j$ with probability $\leq c/m$. Such a decoding algorithm is called a (q, c, ϵ) -smooth decoding algorithm for C .

It may be noted that this definition of a smooth code does not require the decoding procedure to work for corrupted messages. Rather, any particular index $C(x)_j$ is required not to be read overly often in comparison with the other indices.

Later we will show that there is a close relationship between PIR schemes and smooth codes. This in turn establishes a link between PIR and LDCs due to the following fact.

Theorem 2.7 ([13, Theorem 1]). *Let $C : \Gamma^n \rightarrow \Sigma^m$ be a (q, δ, ϵ) -locally decodable code. Then C is also a $(q, q/\delta, \epsilon)$ -smooth code.*

Proof. Let A be a (q, δ, ϵ) -local decoding algorithm for C . For all $i \in [n]$, let S_i be the set of indices $j \in [m]$ for which $\Pr[A(\cdot, i) \text{ reads index } j] > q/\delta m$. Since $A(\cdot, i)$ reads at most q indices, we see that $|S_i| \leq \delta m$.

Define a new algorithm A' as follows. The algorithm $A'(\cdot, i)$ runs $A(\cdot, i)$ as a black box, with $A'(\cdot, i)$ intercepting requests by the $A(\cdot, i)$ for reads to indices of $C(x)$. If the index requested is not in S_i , $A'(\cdot, i)$ simply reads the index and returns the result to $A(\cdot, i)$. If the index is in S_i , however, $A'(\cdot, i)$ simply returns 0. Thus we see that

$$\Pr[A'(C(x), i) = x_i] = \Pr[A(y, i) = x_i],$$

where we define the string y as follows

$$y_j = \begin{cases} 0 & \text{if } j \in S_i \\ C(x)_i & \text{if } j \notin S_i \end{cases}$$

Since $|S_i| \leq \delta m$, we see that $\Delta(C(x), y) \leq \delta m$. Thus the above probabilities are at least $1/2 + \epsilon$, and so we see that A' is a $(q, q/\delta, \epsilon)$ -smooth decoding algorithm for C . \square

There is also a converse translation from a smooth code to an LDC. Take any (q, c, ϵ) -smooth code C . By definition, in decoding a character x_i , any index $C(x)_j$ is read with probability at most c/m . Thus by the union bound we see that

$$\begin{aligned} \Pr[\exists \text{ a corrupted index among the ones we query}] &\leq \sum_{\text{corrupted indices } j} \Pr[j \text{ is read}] \\ &\leq (\delta m)(c/m) = c\delta. \end{aligned}$$

Thus for any δ we have a $(q, \delta, \epsilon - c\delta)$ -locally decodable code C since the probability of reading only non-corrupted indices is at least $1 - c\delta$.

2.3 Private Information Retrieval

Next we define the notion of a private information retrieval protocol.

Definition 2.8. *A one-round, $(1 - \rho)$ -secure, k -server private information retrieval scheme with recovery probability $1/2 + \epsilon$, query size t , and answer size w consists of a randomized algorithm (executed by the user) and k servers S_1, \dots, S_k such that*

- *For some $i \in [n]$ such that the user wishes to retrieve bit x_i from the database, the user generates k t -bit queries q_1, \dots, q_k and sends these to the respective servers S_1, \dots, S_k . The j th server sends back a response of length w equal to the value of some function evaluated on inputs x and q_j .*
- *The user then outputs her guess b for x_i based on i and w_1, \dots, w_k . For all x and i , $\Pr[b = x_i] \geq 1/2 + \epsilon$.*
- *For all x and j , the probability distributions on q_j are ρ -close (in terms of total variation distance) for all different i .*

All randomness is taken over the user's internal coin flips. The scheme is linear if, for every j and query q_j , the j th server's answer w_j is a linear combination over \mathbb{F}_2 of the bits in x .

The communication complexity of the scheme is seen to be $k(t + w)$, i.e. the amount of communication between the user and each server multiplied by the number of servers. The scheme is called one-round because there is exactly one pair of communications from the user to the servers and back again. All known PIR schemes have one round of communication, $\epsilon = 1/2$ (i.e. zero error probability), and $\rho = 0$ (i.e. perfect privacy). We will assume one round and $\rho = 0$ throughout this paper unless explicitly mentioned.

3 Private Information Retrieval Schemes

We begin with an exposition of various PIR schemes, and then show how in general to construct an LDC from a PIR protocol.

3.1 A First Example

In their initial paper on PIR, Chor *et al.* [5] introduced the notion of Private Information Retrieval and presented three PIR schemes:

- A scheme with $O(n^{1/3})$ communication complexity for $k = 2$ servers.
- A scheme with $O(n^{1/k})$ communication complexity for $k \geq 2$ servers.
- A scheme with $O(\log^2 n \log \log n)$ communication complexity for $k = O(\log n)$ servers.

As an example, we illustrate here a simple PIR scheme with a slightly worse $O(n^{1/2})$ communication complexity for $k = 2$ servers. We arrange the n -bit database in a $\sqrt{n} \times \sqrt{n}$ square. Thus the bit x_i is uniquely determined by its coordinates (i_1, i_2) in this square. The user then randomly chooses a \sqrt{n} -bit string y and sends as query q_1 the string y to server S_1 and sends as query q_2 the string $y \oplus e_{i_1}$ to server S_2 . (Here e_{i_1} denotes \sqrt{n} -element vector with all zeros except a 1 at position i_1 .) The servers respond by sending the strings $q_j \cdot C_1, \dots, q_j \cdot C_{\sqrt{n}}$ for $j \in \{1, 2\}$ back, where C_ℓ denotes the ℓ th column of the square database. The user then selects the bits $q_1 \cdot C_{i_2}$ and $q_2 \cdot C_{i_2}$ from the two responses and computes their XOR:

$$(y \cdot C_{i_2}) \oplus ((y \oplus e_{i_1}) \cdot C_{i_2}) = e_{i_1} \cdot C_{i_2} = x_i,$$

and thus the user is able to retrieve the desired bit with perfect privacy and perfect probability.

3.2 The Polynomial Method

The first improvements over the $O(n^{1/k})$ upper bound in the case for a general k servers all made integral use of recursive techniques. This method for constructing PIR schemes involves viewing the database as a multivariate polynomial and applying a recursive scheme which allows one to generate a new PIR protocol P' for $k' > k$ servers from an existing protocol P for k servers. The requirements for P to be suitable for this recursion are similar to those on the 2-server protocol in Ambainis's construction: the majority of the communication must go from the servers to the user, the user must need only a small subset of the bits from the servers' responses, and finally each response must consist of many sub-responses such that each sub-response is known to several servers. Then the user and servers execute a two-level scheme where the servers, rather than sending their long responses back to the user, treat the sub-responses as new database strings. Then the user executes another PIR protocol on these strings in order to retrieve only the bits she requires from them, reducing the total communication. Throughout this section we stay close to the original proofs in [3].

First, we point out that throughout this construction all arithmetic is done over \mathbb{F}_2 . To apply this method then, one associates with every index i of the n -bit database string x an m -bit encoding $E(i)$. We then represent x by an m -variate degree- d polynomial $P_x(Z_1, \dots, Z_m)$ such that the following condition holds:

$$\forall i \in [n], P_x(E(i)) = x_i.$$

The coefficients of P_x depend only on x , and thus each server S_j is able to compute them. The user's goal then is to evaluate $P_x(E(i))$ without giving away to the servers the value of $E(i)$, as this would give away information about the index i . Thus the user randomly chooses j strings \vec{y}_j of length m such that

$$\sum_{j=1}^k \vec{y}_j = E(i)$$

and sends to each server S_j all \vec{y}_ℓ such that $\ell \neq j$. (Throughout this section we use uppercase letters to denote variables and lowercase letters to denote assignments to those variables. The vector notation is used to emphasize the fact that the strings which encode the indices are usually longer than 1 bit long.) Thus the distribution over the strings which each server sees is uniform for all i , and so the servers learn nothing about i from this step.

For the encoding $E(\cdot)$, we simply associate with each index i an m -bit string $E(i)$ such that the Hamming weight $|E(i)| = d$. This is possible if $n \leq \binom{m}{d} \approx \frac{m^d}{d!}$, and as we treat d as a constant this shows that $m = \Theta(n^{1/d})$ variables are sufficient. To facilitate our analyses, we define the following auxiliary polynomial

$$Q(\{Y_{j,h}\}_{j \in [k]}^{h \in [m]}) := P_x\left(\sum_{j=1}^k Y_{j,1}, \dots, \sum_{j=1}^k Y_{j,m}\right) = \left(\sum_{j=1}^k Y_{j,1}\right) \left(\sum_{j=1}^k Y_{j,2}\right) \dots \left(\sum_{j=1}^k Y_{j,d}\right).$$

Thus we construct Q from P simply by setting each $Z_h = \sum_{j=1}^k Y_{j,h}$. Note that Q has k^d monomials each of degree d , with each such monomial of the form $Y_{j_1,1} Y_{j_2,2} \dots Y_{j_d,d}$.

3.2.1 An Example of the Polynomial Method

As a first example of the polynomial method, consider some monomial M of Q - this monomial depends on at most d variables. As each variable is known to $k - 1$ of the servers, then by the pigeonhole principle there exists at least one server which is missing at most $\lfloor d/k \rfloor$ of the variables in M . We assign M randomly to one such server. Consider the case where $d = 2k - 1$. In this case, we assign each monomial M to a server which is missing $\lfloor d/k \rfloor = 1$ variables from M . The servers assigned monomials substitute the values for the variables in their monomials which they know. After this substitution the sum of all such monomials for a given server S_j can be expressed as a degree-1 polynomial $P_j(Y_{j,1}, \dots, Y_{j,m})$ whose variables are those not known to S_j . If the user was able to learn all such polynomials P_j , then by substituting the values for all their variables and summing the values of the polynomials the user would get

$$\sum_{j=1}^k P_j(y_{j,1}, \dots, y_{j,m}) = P_x\left(\sum_{j=1}^k \vec{y}_j\right) = P_x(E(i)) = x_i.$$

The PIR protocol thus consists of the user sending the vectors \vec{y}_j to all servers except S_j , and then each server S_j sends back the $m + 1$ coefficients (each a single bit) of the degree-1 polynomial P_j . The complexity of the protocol is thus $O(m) = O(n^{1/d}) = O(n^{1/(2k-1)})$.

3.2.2 Beimel's Recursive Construction

Next, we present Beimel's more complicated construction which yields a slightly better upper bound. First we define the new polynomial P_x :

$$P_x(Z_1, \dots, Z_m) := \sum_{i=1}^n x_i \prod_{E(i)_\ell=1} Z_\ell.$$

We note that since each string $E(i)$ has Hamming weight d , the polynomial P_x is the sum of monomials each of degree d and thus the degree of P_x is also d as desired. Further, when we evaluate P_x on the encoding of some index $E(i)$ (by treating the ℓ th bit position of $E(i)$ as the assignment to Z_ℓ), the only monomial whose value is 1 is that whose coefficient is x_i . Thus we see that $P_x(E(i)) = x_i$. Note that these monomials are multilinear monomials, as we're working in \mathbb{F}_2 and thus for $a \geq 1$, $x^a = x$.

In the recursive protocol, we assume that each server's response consists of several sub-responses, and that each sub-response is known to several servers. The recursion involves viewing these sub-responses as new database strings and in turn representing these strings by polynomials. We thus want to show how to construct a set of polynomials such that

$$\sum_{j=1}^k P_x(\vec{y}_j) = \sum_{V \subseteq [k]} P_V(z_v),$$

where P_V is a polynomial computable by every server in the set $\{S_j \mid j \in V\}$, and z_v is some assignment to this polynomial known to the user. We now define two parameters: λ and k' . Let k' be the minimal number of servers in any set V to which we assign a polynomial P_V . Further, the polynomials P_V consist of sums of monomials M . Let λ be the maximum number of variables each of the servers in V misses from each of the monomials in P_V . To complete our protocol, we will show how to construct polynomials P_V and P_j such that for every i

$$x_i = P_x(E(i)) = \sum_{V \subseteq [k], |V| \geq k'} P_V(E(i)) + \sum_{j=1}^k P_j(\vec{y}_j).$$

In addition, each P_V can be computed from P_x and $\{\vec{y}_\ell\}_{\ell \notin V}$ and each P_j can be computed from P_x and $\{\vec{y}_\ell\}_{\ell \neq j}$.

Before we show how to construct the polynomials P_V and P_j , we first prove a small lemma which gives an upper bound on d which guarantees that for any monomial M of degree at most d , there either exists a server S_j which knows all but at most one variable in M (we will then include M in the polynomial P_j), or the number of servers which miss at most λ variables in M is greater than or equal to k' .

For some monomial M in the variables $Y_{j,h}$, let $V(M) \subseteq [k]$ denote the set of servers which each have corresponding variables that appear at most λ times in M . (By a corresponding variable for server S_j , we mean a variable $Y_{j,\ell}$ such that $j_\ell = j$.) Now we prove the following lemma.

Lemma 3.1. *Let k be the number of servers, and for parameters λ, k' , and d , let $k' \leq k$ and $d \leq (\lambda + 1)k - (\lambda - 1)k' + (\lambda - 2)$. Further, let M be a monomial of degree at most d in the variables $Y_{j,h}$, with $j \in [k]$ and $h \in [m]$. Then either there is a server which misses at most one variable from M , or $|V(M)| \geq k'$.*

Proof. Assume toward contradiction that the lemma does not hold. Thus every server S_j misses at least two variables in M (and thus at least two variables of the form $Y_{j,h}$ appear in M) and

at most $k' - 1$ servers are missing at most λ variables from M , which restated means that at least $k - (k' - 1)$ servers are missing at least $\lambda + 1$ variables. Thus we see that the number of variables in M is at least

$$\begin{aligned} (k - (k' - 1))(\lambda + 1) + 2(k' - 1) &= (\lambda + 1)k - \lambda k' + \lambda - k' + 1 + 2k' - 2 \\ &= (\lambda + 1)k - (\lambda - 1)k' + (\lambda - 2) + 1 \\ &\geq d + 1, \end{aligned}$$

which contradicts our assumption about d , the maximum degree of M . \square

We now show via the following lemma how to construct the polynomials P_V and P_j .

Lemma 3.2. *Let k, λ, k' , and d be as in Lemma 3.1 and $P_x(Z_1, \dots, Z_m)$ be a polynomial of degree at most d . Then there are polynomials $P_V(Z_1, \dots, Z_m)$ for every $V \subseteq [k]$, where $|V| \geq k'$, and polynomials $P_j(Z_1, \dots, Z_m)$ for $j \in [k]$ such that*

1. Each polynomial P_V is of degree $\lambda|V|$ and can be computed from P_x and $\{\vec{y}_\ell\}_{\ell \notin V}$,
2. Each polynomial P_j is of degree 1 and can be computed from P_x and $\{\vec{y}_\ell\}_{\ell \neq j}$,
3. $P_x(E(i)) = \sum_{V \subseteq [k], |V| \geq k'} P_V(E(i)) + \sum_{j=1}^k P_j(\vec{y}_j)$.

Proof. It is sufficient to prove the lemma for polynomials P'_x which consist of only a single monomial, as we can then just sum over all such monomials in the polynomial P_x . Thus we prove the Lemma without loss of generality for the polynomial $P(Z_1, \dots, Z_m) = Z_1 Z_2 \dots Z_d$, i.e. a polynomial consisting of a single monomial with the maximum allowed degree d .

Let

$$T(M) := \prod_{j_q \in V(M)} Z_q \prod_{j_q \notin V(M)} Y_{j_q, q}.$$

Note that $T(M)$ contains terms both of the form Z_q and of the form $Y_{j_q, q}$. The intention is that, if $T(M)$ is included as part of a polynomial in the variables $\{Y_{j, h}\}$, then each of the Z_q terms should be expanded to $\sum_{j=1}^k Y_{j, q}$. If instead $T(M)$ is included as part of a polynomial in the variables $\{Z_q\}$, then it should be interpreted as a single monomial. In this case, since the coefficient $\prod_{j_q \notin V(M)} Y_{j_q, q}$ is known to all servers in $V(M)$ these servers can substitute the coefficient in the monomial. Thus the monomial's degree is $\leq \lambda|V(M)|$, as each server appears in $V(M)$ at most λ times.

We now give an algorithm to construct the required polynomials P_V . A sketch of its operation is as follows. The algorithm handles monomials M with sets $V(M) \geq k'$ one at a time, placing them in the polynomials P_V where $V = V(M)$. Monomials may be removed from this set and added again throughout the running of the algorithm, but care is taken to ensure that the algorithm eventually halts after having handled all necessary monomials and thus having constructed the requisite polynomials P_V .

Let $r(M)$ denote the number of variables $Y_{j_q, q}$ in M with $j_q \in V(M)$. The algorithm executes the following steps:

1. Set $Q' = Q$ and for all V set $P_V(Z_1, \dots, Z_m) = 0$.
2. Find the size of the largest set $V(M)$ for any of the monomials M currently in Q' . Set V equal to one of these sets $V(M)$. If $|V| < k'$ then STOP.

3. While there is a monomial M such that $V(M) = V$:
 - Among these M 's choose an M that maximizes $r(M)$.
 - Add $T(M)$ to P_V and set $Q' = Q' - T(M)$.

4. GOTO 2.

To see that the algorithm produces the desired polynomials P_V , view each $T(M)$ added to them in Step 3 as a sum of monomials in the variables $\{Y_{j,h}\}$:

$$T(M) = \prod_{j_q \notin V(M)} Y_{j_q,q} \prod_{j_q \in V(M)} \left(\sum_{j=1}^k Y_{j,q} \right).$$

If we expand the second product, we see that there are exactly $k^{r(M)}$ monomials in each $T(M)$, each of degree d . Further, since $Q = P$ and since we construct the polynomials P_V by simultaneously adding a $T(M)$ to some P_V and subtracting this $T(M)$ from Q' , we see that at the end of the algorithm that

$$P(Z_1, \dots, Z_m) = \sum_{V \subseteq [k], |V| \geq k'} P_V(Z_1, \dots, Z_m) + Q'(Z_1, \dots, Z_m).$$

Now, note that the only monomials M in Q' when the algorithm halts are those for which $|V(M)| < k'$, and thus by Lemma 3.1 and our choice of d, λ , and k' we know that for each such M there must exist some server which only misses at most one variable from M . Thus we assign each monomial M in Q' to a polynomial P_j such that server S_j misses at most one variable from M .

The last thing necessary is an argument which shows that the algorithm eventually halts. We define an equivalence relation over monomials $M_1 = Y_{j_1,1} Y_{j_2,2} \dots Y_{j_d,d}$ and $M_2 = Y_{j'_1,1} Y_{j'_2,2} \dots Y_{j'_d,d}$ as follows. We say that $M_1 \equiv M_2$ with respect to some subset $W \subseteq [k]$ if the following two conditions are met: (1) $V(M_1) = V(M_2) = W$ and (2) for each index $q \in [d]$, either j_q, j'_q are both in W or $j_q = j'_q$ (and thus the same variable $Y_{j_q,q}$ appears in both monomials).

Now let M_1 be a monomial of $T(M)$ and note the following:

1. $V(M_1) \subseteq V(M)$, because for any server not to be in $V(M)$ it must appear in M more than λ times, and the variables so included in M are also included in M_1 due to the way we define $T(M)$. Thus these servers also appear in M_1 at least the same number of times.
2. $r(M_1) \leq r(M)$. Due to how we construct $T(M)$, the maximum number of variables in M_1 which correspond to servers in $V(M_1)$ is $|V(M)|$.
3. If $V(M_1) = V(M)$ and $r(M_1) = r(M)$, then by definition $M_1 \equiv M$ (with respect to $V(M)$).
4. If $M_2 \equiv M_1$ (with respect to some set W), then M_2 must also be in $T(M)$ because by (1) $V(M_2) = W \subseteq V(M)$ and the way we have constructed $T(M)$ ensures that all monomials for which this is true are included in $T(M)$.

Thus we see that if $M_1 \equiv M_2$ then, at any stage of the algorithm, either both M_1 and M_2 are in Q' or both of them are not. This is because it is true at the beginning (as we set $Q' = Q$, which is simply the sum of *all* monomials of the form $Y_{j_1,1} Y_{j_2,2} \dots Y_{j_d,d}$) and whenever we update Q' by subtracting $T(M)$ for some monomial M , by (4) we either add both M_1 and M_2 to Q' or subtract them both. (Note that since we are working in \mathbb{F}_2 , subtraction and addition amount

to the same thing. By removing a monomial, we mean adding it twice so that its coefficient becomes even and thus equal to $0 \pmod{2}$.)

In order to see now that the algorithm halts, note that even though new monomials M' may be added to Q' by subtracting $T(M)$ throughout the running of the algorithm, these monomials always either have smaller $V(M')$ or $r(M')$ and thus we are always progressing. To see this, remember that we always pick in Step 2 a maximal set V and among the monomials M for which $V(M) = V$, we always choose one in Step 3 such that it maximizes $r(M)$. We thus delineate three cases which characterize the monomials which are added to Q' when we subtract $T(M)$:

1. $V(M') \subsetneq V(M)$. In this case M' will simply be dealt with at a future point in the algorithm when $V(M')$ is chosen in Step 2 (provided it isn't removed before then by a subsequent addition of another $T(M)$).
2. $V(M') = V(M)$ but $r(M') < r(M)$. In this case M' will be dealt with at a future point in the loop of Step 3, provided again that it hasn't been removed before then.
3. $V(M') = V(M)$ and $r(M') = r(M)$. In this case, by (4) above, when we subtract $T(M)$ from Q' we remove both M and M' from Q' . After we finish Step 3, we've completed the construction of the polynomial P_V and thus we won't return to either monomial again.

Thus we see that the algorithm is both correct and that it halts, and so how we can construct the desired polynomials. \square

Finally, let us analyze the communication complexity of the resultant PIR protocol. Note that, even though we are implementing a recursive scheme, we can still implement the protocol in 1-round by having the user send simultaneously her queries for both the k -server protocol, which determines the database strings to be used in the latter protocols, and the k' -server protocols at once. The servers then respond in a single round with the coefficients required by the user to evaluate the polynomials P_V and P_j . Thus the communication complexity is equal to the sum of the communication required to retrieve the necessary information about the sets of both the polynomials P_V and P_j . The communication required to retrieve the polynomials P_j is simply $O(m) = O(n^{1/d})$, as these polynomials are degree-1 polynomials of m variables. Thus to retrieve them requires each of their $m+1$ coefficients to be transmitted. To retrieve the necessary coefficients for the polynomials P_V , we execute with each set of servers V such that $|V| \geq k'$ a different protocol P' as described above. Now, let $C_P(n, k)$ denote the complexity for a protocol P with database length n and k -servers. Thus we have the following theorem.

Theorem 3.3. *Given a PIR protocol P with communication complexity $C_P(n, k)$ and positive integers d, λ, k' such that $k' < k$ and $d \leq (\lambda + 1)k - (\lambda - 1)k' + (\lambda - 2)$, there exists a PIR protocol P' with communication complexity*

$$C_{P'}(n, k) = O\left(n^{1/d} + \sum_{\ell=k'}^k \binom{k}{\ell} C_P(n^{\lambda\ell/d}, \ell)\right) = O\left(n^{1/d} + k \binom{k}{k'} C_P(n^{\lambda k'/d}, k')\right).$$

The second equality is justified by the fact that in this protocol, the complexity cannot increase by adding more servers. Thus complexity of the sum is upper bounded by the complexity of the term with the smallest number of servers. In general, this quantity is dominated by the C_P term which comes from communicating the coefficients of the P_V polynomials, as should be expected, and so we drop the $n^{1/d}$ from the analysis below. In addition, we generally treat k (and thus k') as a constant, so the $k \binom{k}{k'}$ factor can be dropped from the analysis as well. This gives a recursive complexity analysis - let us analyze this so as to give a specific bound.

Lemma 3.4. *For every integer $i \geq 3$, there is a PIR protocol P_i such that $C_{P_i}(n, k) = O(n^{2/(ik)})$ for every constant $k \geq (i-1)!$, where the constant depends on k .*

Proof. As shown above, there exists a protocol that achieves $O(n^{1/(2k-1)})$ communication for all k . Thus for $i = 3$, we require a protocol of $O(n^{1/(1.5k)})$ but already have a better one, and we also have the necessary protocols for $i = 1, 2$. For the induction hypothesis, assume that $i \geq 3$ and that there exists a PIR protocol P_i such that $C_{P_i}(n, k) = O(n^{2/(ik)})$ for every $k \geq (i-1)!$. Thus we need to show how to construct a protocol P' such that $C_{P'}(n, k) = O(n^{2/((i+1)k)})$ for all $k \geq i!$. To do this, set $k' = \lfloor \frac{k}{i} \rfloor$ (and so $k' \geq (i-1)!$ since $k \geq i!$), set $\lambda = \lceil \frac{i}{2} \rceil$ (and so $\lambda \geq 2$, since $i \geq 3$), and set $d = (\lambda + 1)k - (\lambda - 1)k'$ (and so $d \leq (\lambda + 1)k - (\lambda - 1)k' + (\lambda - 2)$, since $\lambda \geq 2$). Thus we can apply Theorem 3.3, which states that there exists a protocol P_{i+1} of complexity $C_{P_{i+1}}(n, k) = C_{P_i}(n^{\lambda k'/d}, k') = O(n^{2\lambda k'/(idk')}) = O(n^{2\lambda/(id)})$. Thus we need to show that $2\lambda/(id) \leq 2/((i+1/k))$, which we do as follows

$$\frac{\lambda}{id} \leq \frac{\lambda}{i((\lambda + 1)k - (\lambda - 1)k'/i)} = \frac{\lambda}{k(\lambda i + (i - \lambda + 1))} \leq \frac{\lambda}{k(\lambda i + \lambda)} = \frac{1}{k(i + 1)},$$

where the first inequality is justified by a substituting the assumed bound on d and substituting k/i for k' , while the second inequality is justified by the fact that $\lambda = \lceil \frac{i}{2} \rceil$. \square

We are now ready to prove the main result of this section.

Theorem 3.5. *There exists a PIR protocol P with $C_P(n, k) = O(n^{2 \log \log k / (k \log k)})$ for all $k \geq 3$.*

Proof. Set P to be the protocol from Lemma 3.4, and set $i = \lceil \log k / (\log \log k) \rceil$. Thus $(i-1)! \leq (i-1)^{i-1} \leq \log k^{\log k / (\log \log k)} = k$, and so $C_P(n, k) = C_{P_i}(n, k) = O(n^{2/(id)}) = O(n^{2/(k \log k / (\log \log k))}) = O(n^{2 \log \log k / (k \log k)})$. \square

It should be noted that the above bound is general for all k , but that better bounds are possible by choosing optimal values for the parameters in Theorem 3.3 for the specific value of k in which one is interested. As an example, consider the case where $k = 3$. We know there is a 2-server protocol with complexity $O(n^{1/3})$, and we can set $\lambda = k' = 2$ and $d = 7$ (as $(\lambda + 1)k - (\lambda - 1)k' + (\lambda - 2) = 9 - 2 = 7$). Thus we apply Theorem 3.3 to generate a protocol with complexity $O(n^{1/7} + n^{(2 \cdot 2)/(7 \cdot 3)}) = O(n^{4/21})$.

4 LDC Constructions

In this section we discuss known constructions of LDCs, with the main focus being Yekhanin's construction based on Regular Intersecting Families from [29, 14]. Before the publication of this result, all of the best known LDC constructions actually came from translations of PIR schemes. We thus begin the discussion of LDCs by showing how to perform such a translation.

4.1 Getting Smooth Codes from PIR Schemes

In this section we establish, via a lemma from [10], a connection between smooth codes and PIR schemes. The intuition should be clear - one can think of the queries to the indices in a smooth code as particular queries sent to different servers, with the privacy requirement of PIR being analogous to the even distribution requirement of smooth codes. Here we generalize the original proof, which focused on the two-server case, to an arbitrary k servers.

Lemma 4.1. *Assume the existence of a PIR scheme with k servers, database size n , query size t , answer size a , and recovery probability at least $1/2 + \epsilon$. Then there exists a $(k, k+1, \epsilon)$ -smooth code $C : \{0, 1\}^n \rightarrow (\{0, 1\}^a)^m$, with $m \leq (k^2 + k) \cdot 2^t$. Further:*

1. If the PIR scheme is linear, then C is also linear.
2. If in the PIR scheme the user only uses k predetermined bits from each answer a_j , then this is also true for the decoding algorithm of C .

Proof. The intuition is as follows. If we enumerate all possible answers from each server, the PIR system can be viewed as an encoding of the database $x \in \{0, 1\}^n$ as a string $PIR(x) \in (\{0, 1\}^a)^\ell$, where $\ell = k \cdot 2^t$. The user can then reconstruct a bit x_i with probability $1/2 + \epsilon$ by examining k entries of the codeword $PIR(x)$. Let $p_j(i)$ be the probability that when decoding bit x_i the algorithm reads index $PIR(x)_j$. We note that $p_j(i)$ is independent of i due to the privacy condition of the PIR scheme, and so we write simply p_j for the probability that index $PIR(x)_j$ is queried during the decoding of *any* bit x_i . We will use this to guarantee the smoothness property of smooth codes by sacrificing code length somewhat.

Note that $\sum_j p_j \leq k$. We copy entry j of the encoding $n_j = \lceil p_j \cdot \ell \rceil$ times and label the new string with such repetitions $C(x)$. The decoding algorithm for x_i from $C(x)$ generates queries j_1, \dots, j_k just as the algorithm which decodes x_i from $PIR(x)$. However, in addition the algorithm selects at random one of the n_{j_k} copies of each entry j_k to query. If for some entry j we have $p_j \leq 1/\ell$ then $n_j = 1$ and the algorithm simply selects this entry with probability $p_j \leq 1/\ell$. Otherwise the j th entry is copied $n_j = \lceil p_j \cdot \ell \rceil > 1$ times, and each copy is queried with probability p_j/n_j which is

$$\frac{p_j}{\lceil p_j \cdot \ell \rceil} \leq \frac{p_j}{p_j \cdot \ell} = \frac{1}{\ell}.$$

Now the length m of the string $C(x)$ can be derived as follows

$$\begin{aligned} m &= \sum_{j=1}^{\ell} n_j = \sum_{j=1}^{\ell} \lceil p_j \cdot \ell \rceil \\ &\leq \sum_{j=1}^{\ell} (1 + p_j \cdot \ell) = \ell + \sum_{j=1}^{\ell} p_j \cdot \ell \\ &\leq (k + 1) \cdot \ell. \end{aligned}$$

Using this we see that $1/\ell \leq (k + 1)/m$, and so no entry is queried with probability greater than $(k + 1)/m$. Further, the recovery probability of the bit x_i is unchanged by the translation, and thus remains $1/2 + \epsilon$. \square

As an example of the application of this lemma, consider the PIR scheme discussed in Section 3.1 where we have 2 servers, $t = \sqrt{n}$ size queries, $a = \sqrt{n}$ size answers, and $\epsilon + 1/2 = 1$ recovery probability. Thus by the lemma, we know that there exists a $(2, 3, \epsilon)$ -smooth code $C : \{0, 1\}^n \rightarrow (\{0, 1\}^a)^m$ and $m \leq 6 \cdot 2^t$, where the codeword length is large enough so as to allow distinct codewords for all possible combinations of the user's queries and servers' responses. In turn, by the discussion which follows Theorem 2.7, we know that C is also $(2, \delta, \epsilon - 3\delta)$ -locally decodable for all δ .

The best asymptotic codelength for LDCs known is still $m = 2^{O(\log \log k / (k \log k))}$, originally attained by the LDCs derived from the PIR scheme presented in Theorem 3.5. The constant in this bound was greatly improved however by replacing the $O(n^{1/3})$ 2-server base scheme used in the recursion with the $O(n^{1/32576258})$ 3-server scheme of Yekhanin's.

4.2 Yekhanin's Construction

In [29], Yekhanin presented a novel construction for LDCs which greatly improved the code length over any known 3-query LDC. Unlike previous constructions, this one uses algebra over finite fields to craft certain families of sets. The outline of his construction is as follows. The first step is the definition of a certain combinatorial object, viz. that of a *regularly intersecting family* of sets. He then shows that the existence of such a regularly intersecting family implies the existence of an LDC. Next he defines two further concepts - that of a set being *combinatorially nice* and that of a set being *algebraically nice* - and shows how to construct regular intersecting families from sets with both types of niceness. Finally, he shows how to construct such a nice set, and thus an LDC, from any Mersenne prime.

First, we state Yekhanin's main result.

Theorem 4.2 ([29]). *For every positive integer n there exists a code of length $2^{O(n^{1/32582657})}$ which is $(3, \delta, 1/2 - 6\delta)$ -locally decodable for all δ .*

4.2.1 Constructing Regular Intersecting Families from Nice Sets

In this section we define some set-theoretic properties and show how we can construct sets with these properties which imply the existence of LDCs. The first such property applies to families of sets which we call Regular Intersecting Families.

Definition 4.3. *For $i \in [n]$ and $r \in [R]$, let T_i and Q_{ir} be subsets of $[N]$. The subsets T_i and Q_{ir} form a (q, n, N, R, s) **Regular Intersecting Family** if the following conditions are met:*

1. q is odd
2. For all $i \in [n]$, $|T_i| = s$
3. For all $i \in [n]$ and $r \in [R]$, $|Q_{ir}| = q$
4. For all $i \in [n]$ and $r \in [R]$, $Q_{ir} \subseteq T_i$
5. For all $i \in [n]$ and $w \in T_i$, $|\{r \in [R] \mid w \in Q_{ir}\}| = Rq/s$
6. For all $i \neq j \in [n]$ and $r \in [R]$, $|Q_{ir} \cap T_j|$ is even.

This amounts to the following. We divide the set $[N]$ into n (possibly overlapping) sets T_i each of size s . We then further divide these sets T_i into R sets Q_{ir} , each of size q , such that (1) every element of T_i is included in the same number of sets Q_{ir} and (2) the size of the intersection of the subsets $Q_{ir} \subseteq T_i$ with all other sets T_j such that $i \neq j$ is even.

In the following theorem we see that the existence of such set families implies the existence of LDCs.

Theorem 4.4. *The existence of a (q, n, N, R, s) regular intersecting family implies the existence of a $(q, \delta, 1/2 - \delta Nq/s)$ -Locally Decodable Code which encodes n bits to N bits.*

Proof. We identify subsets of N with their incidence vectors - i.e. we map each subset $S \subseteq N$ to a vector $I(S) \in \{0, 1\}^N$ where the j th entry of $I(S)$ is 1 if $j \in S$ and is 0 otherwise. We define the code C by its generator matrix $G \in \{0, 1\}^{n \times N}$ where each row i is defined to be the incidence vector of T_i :

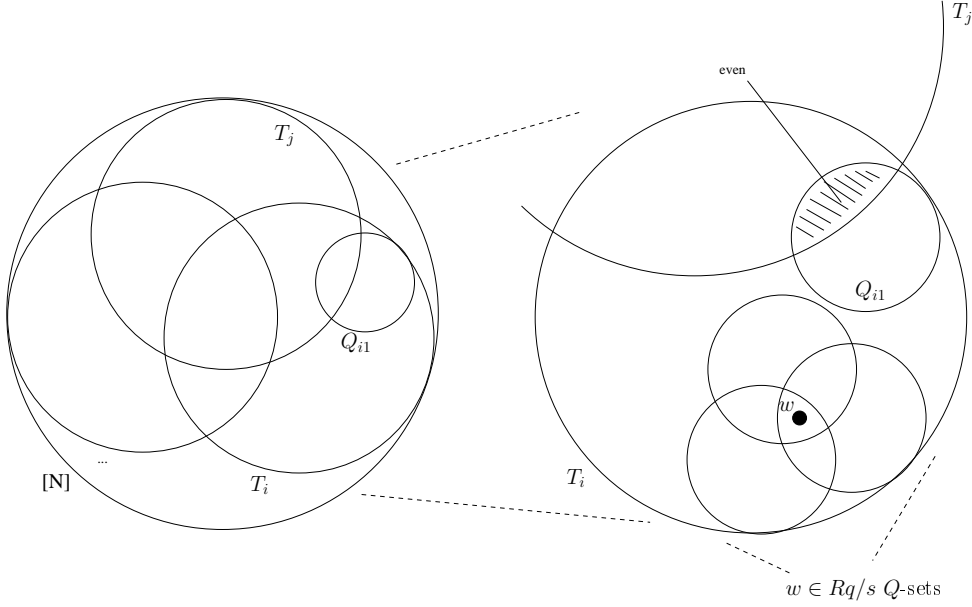


Figure 1: Regular Intersecting Family

$$G = \begin{pmatrix} I(T_1) \\ I(T_2) \\ \dots \\ I(T_n) \end{pmatrix}.$$

Thus a codeword $C(x)$ is given by the product $xG = C(x)$.

Next we define the decoding algorithm A for C as follows. To decode the bit x_i , A first picks a random $r \in [R]$. Next, A makes $q = |Q_{ir}|$ queries to $C(x)$ in order to acquire the entries of Q_{ir} and then outputs their parity. This amounts to outputting the dot product $\langle C(x), I(Q_{ir}) \rangle$ of $C(x)$ and the incidence vector of Q_{ir} . If none of these q bits have been corrupted we note that

$$\langle C(x), I(Q_{ir}) \rangle = \langle xG, I(Q_{ir}) \rangle = \sum_{j=1}^n x_j \langle I(T_j), I(Q_{ir}) \rangle = \sum_{j=1}^n x_j |T_j \cap Q_{ir}| = x_i |Q_{ir}| = x_i.$$

The step from the dot product $\langle xG, I(Q_{ir}) \rangle$ to the sum is justified by the definition of G , and the next step is justified from the fact that we are working in \mathbb{F}_2 . The next equality is justified from Part 6 of Definition 4.3, from which we know for $i \neq j$ that $|T_j \cap Q_{ir}| = 0 \pmod{2}$. The final equality comes from the facts that we know $|Q_{ir}|$ to be odd and that we know $Q_{ir} \subseteq T_i$.

Finally, we know that the probability that some particular index j is in the set Q_{ir} is q/s by Part 5 of Definition 4.3. Thus by the union bound, the probability that one of δN corrupted indices lies in Q_{ir} is $\leq \delta Nq/s$, and so we see that

$$\Pr[A(C(x), i) = x_i] \geq 1 - \delta Nq/s = 1/2 + (1/2 - \delta Nq/s).$$

□

Next we define two separate properties of sets which together imply the existence of regular intersecting families. Recall that \mathbb{F}_p^* denotes the multiplicative subgroup of \mathbb{F}_p , whose elements are all non-zero elements of \mathbb{F}_p .

Definition 4.5. A set $S \subseteq \mathbb{F}_p^*$ is called (m, n) combinatorially nice if there exist vectors $\{u_1, \dots, u_n\}$ and $\{v_1, \dots, v_n\} \in \mathbb{F}_p^m$ such that

1. $\langle u_i, v_i \rangle = 0$.
2. For all $i \neq j$, $\langle u_j, v_i \rangle \in S$.

Note that (2) implies that $\langle u_j, v_i \rangle \neq 0$, since $0 \notin \mathbb{F}_p^m$.

Definition 4.6. A set $S \subseteq \mathbb{F}_p^*$ is called q algebraically nice if q is odd and there exist two sets $S_0, S_1 \subseteq \mathbb{F}_p$ such that

1. $|S_0| > 0$.
2. $|S_1| = q$.
3. For all $\alpha \in \mathbb{F}_p$ and $\beta \in S$, $|S_0 \cap (\alpha + \beta S_1)|$ is even.

We now prove that a set possessing both properties gives rise to a regular intersecting family.

Theorem 4.7. A set $S \subseteq \mathbb{F}_p^*$ which is both (m, n) combinatorially nice and q algebraically nice gives rise to a $(q, n, p^m, |S_0|p^{m-1}, |S_0|p^{m-1})$ -regular intersecting family, where S_0 and S_1 are the sets from the definition of algebraic niceness.

Proof. Let $N = p^m$ and $R = s = |S_0|p^{m-1}$. Assume some bijection between $[N]$ and \mathbb{F}_p^m . For all $i \in [n]$, let u_i, v_i be the vectors from Definition 4.5 and further let

$$T_i = \{x \in \mathbb{F}_p^m \mid \langle u_i, x \rangle \in S_0\}.$$

Thus we can see that $|T_i| = |S_0|p^{m-1} = R$ as follows. Build a vector x up one position at a time ensuring that x satisfies the criterion that $\langle u_i, x \rangle \in S_0$. For each of the first $m - 1$ positions, we have complete freedom in choosing any of the p elements from \mathbb{F}_p . However, for the last position, we're constrained to choose an element such that the dot product is in S_0 , and there are obviously exactly $|S_0|$ elements for which this is the case. Thus the total number of vectors x is $|S_0|p^{m-1}$. Now assume for each i some bijection between between $[R]$ and $T_i = \{w_{i1}, \dots, w_{is}\}$. Let

$$Q_{ir} = \{w_{ir} + \lambda v_i \mid \lambda \in S_1\}.$$

Now let us verify that the sets T_i and Q_{ir} satisfy the conditions of regular intersecting families. The first condition states that $|Q_{ir}| = q$ must be odd, and this is guaranteed by Definition 4.6 of algebraic niceness. The second condition requires that for all $i \in [n]$, $|T_i| = s$, which is true by assumption. The third condition, which states that for all $i \in [n]$, $r \in [R]$, $|Q_{ir}| = q$, is again true by assumption.

For condition 4, note that by Definition 4.5 we know that $\langle u_i, v_i \rangle = 0$. Thus for every $\lambda \in S_1$ we see that $\langle u_i, w_{ir} + \lambda v_i \rangle = \langle u_i, w_{ir} \rangle + \lambda \langle u_i, v_i \rangle = \langle u_i, w_{ir} \rangle$, which we know to be in T_i from the definition of T_i . Therefore $Q_{ir} \subseteq T_i$.

For condition 5, note that for any fixed i and $w \in T_i$ we have

$$\begin{aligned}
|\{r \in [R] \mid w \in Q_{ir}\}| &= |\{w_{ir} \in T_i \mid \exists \lambda \in S_1, w = w_{ir} + \lambda v_i\}| \\
&= |\{w_{ir} \in T_i \mid \exists \lambda \in S_1, w_{ir} = w - \lambda v_i\}| \\
&= |S_1| = q,
\end{aligned}$$

with the third equality justified by the fact that for every $\lambda_1 \neq \lambda_2 \in S_1$, $w - \lambda_1 v_i \neq w - \lambda_2 v_i$ as $v_i \neq 0$. Thus the number of distinct such vectors is simply equal to the size of S_1 . Since $R = s$, we see that $Rq/s = q$, which is precisely what condition 5 requires.

For condition 6, fix $i, j \in [n]$ and $r \in [R]$ such that $i \neq j$. Note that

$$\begin{aligned}
|Q_{ir} \cap T_j| &= |\{w_{ir} + \lambda v_i \mid \lambda \in S_1\} \cap \{x \in \mathbb{F}_p^m \mid \langle u_j, x \rangle \in S_0\}| \\
&= |\{\lambda \in S_1 \mid \langle u_j, w_{ir} + \lambda v_i \rangle \in S_0\}| \\
&= |\{\lambda \in S_1 \mid (\langle u_j, w_{ir} \rangle + \lambda \langle u_j, v_i \rangle) \in S_0\}| \\
&= |S_0 \cap (\langle u_j, w_{ir} \rangle + \langle u_j, v_i \rangle S_1)|.
\end{aligned}$$

The first equality follows from the definitions of Q_{ir} and T_j . The last equality follows from the fact that $\langle u_j, v_i \rangle \in S$, which implies that $\langle u_j, v_i \rangle \neq 0$. This quantity is seen to be even by Part 3 of Definition 4.6. \square

4.2.2 Combinatorially Nice Sets

We start off the discussion of combinatorially nice sets with a review of tensor algebra.

Definition 4.8. For vectors $w \in \mathbb{F}_p^m$ and $x \in \mathbb{F}_p^n$, define the **Tensor Product** $w \otimes x$ as follows

$$w \otimes x = (w_1 x_1, w_1 x_2, \dots, w_1 x_n, w_2 x_1, \dots, w_m x_n).$$

Note that $\dim(w \otimes x) = \dim(w) \times \dim(x)$.

Definition 4.9. For a vector $w \in \mathbb{F}_p^m$, define the ℓ th **Tensor Power** $w^{\otimes \ell}$ of w as follows

$$w^{\otimes \ell} = \underbrace{w \otimes w \otimes \dots \otimes w}_{\ell \text{ times}}.$$

Note that $\dim(w^{\otimes \ell}) = \dim(w)^\ell$.

Now, for $w \in \mathbb{F}_p^m$ and positive integer ℓ , consider $w^{\otimes \ell} \in \mathbb{F}_p^{m^\ell}$ and label the coordinates of $w^{\otimes \ell}$ by all possible sequences in $[m]^\ell$ such that $w_{i_1, \dots, i_\ell}^{\otimes \ell} = \prod_{j=1}^\ell w_{i_j}$.

Lemma 4.10. Let p be an odd prime, $z \geq p - 1$ be a positive integer, and S be a subgroup of \mathbb{F}_p^* . Then S is $\left(\binom{z-1+(p-1)/|S|}{(p-1)/|S|}, \binom{z}{p-1}\right)$ combinatorially nice.

Proof. Let $n = \binom{z}{p-1}$. For $i \in [n]$ let vectors u_i'' be the incidence vectors of all $(p-1)$ -subsets of $[z]$ and vectors v_i'' be their complements. Thus $\langle u_i'', v_i'' \rangle = 0$ and $\langle u_i'', v_j'' \rangle \neq 0$ for $i \neq j$. Let $\ell = (p-1)/|S|$ (which is an integer due to Euler's theorem [12, Theorem 1.7]) and u, v be vectors in \mathbb{F}_p^z . Then

$$\begin{aligned} \langle u^{\otimes \ell}, v^{\otimes \ell} \rangle &= \sum_{\langle i_1, \dots, i_\ell \rangle \in [z]^\ell} \left(\prod_{j=1}^{\ell} u_{i_j} \prod_{j=1}^{\ell} v_{i_j} \right) = \\ &= \sum_{\langle i_1, \dots, i_\ell \rangle \in [z]^\ell} \left(\prod_{j=1}^{\ell} u_{i_j} v_{i_j} \right) = \left(\sum_{i_1 \in [z]} u_{i_1} v_{i_1} \right) \dots \left(\sum_{i_\ell \in [z]} u_{i_\ell} v_{i_\ell} \right) = \langle u, v \rangle^\ell. \end{aligned}$$

For $i \in [n]$, let $u'_i = u_i''^{\otimes \ell}$ and $v'_i = v_i''^{\otimes \ell}$. The above formula along with the cyclicity of \mathbb{F}_p^* imply that both $\langle u'_i, v'_i \rangle = 0$ for all $i \in [n]$ and $\langle u'_j, v'_i \rangle \neq 0$ for $i \neq j \in [n]$.

Note that the entries in the vectors u'_i and v'_i at (i_1, \dots, i_ℓ) and (i'_1, \dots, i'_ℓ) are equal provided that the multisets $\{i_1, \dots, i_\ell\}$ and $\{i'_1, \dots, i'_\ell\}$ are equal. Thus these vectors contain many repeated entries of the same value associated with a particular multiset. A basic theorem of combinatorics states that the number of integer solutions to the equation $x_1 + x_2 + \dots + x_z = \ell$, with $x_i \geq 0$ for all i , is equal to $\binom{z+\ell-1}{\ell}$ [12, Proposition 1.4], and thus the number of distinct multisets with elements from $[z]$ of cardinality ℓ is $s = \binom{z-1+\ell}{\ell}$. We construct smaller vectors of length s by counting the number of repetitions for any particular multiset. For each u_i , we collapse all repeated entries for a particular multiset to a single entry equal to the number of repetitions. For each v_i , we instead collapse all repeated entries for a particular multiset to the value which was repeated. Finally, note that this preserves the inner product: $\langle u_i, v_i \rangle = \langle u'_i, v'_i \rangle$. \square

Per a suggestion from Ronald de Wolf [27], we observe, using the following fact, that this construction is nearly optimal.

Fact 4.11 ([12, Lemma 14.11]). *For $i = 1, \dots, n$ let $f_i : \Omega \rightarrow \mathbb{F}$ be functions and $v_i \in \Omega$ elements such that*

- $f_i(v_i) \neq 0$ for all $1 \leq i \leq n$
- $f_i(v_j) = 0$ for all $1 \leq j < i \leq n$.

Then the set $\{f_1, \dots, f_m\}$ is linearly independent in the space \mathbb{F}^Ω .

Theorem 4.12. *For any (m, n) -combinatorially nice subgroup $S \subseteq \mathbb{F}_p^*$, the following inequality holds*

$$n \leq \binom{m + |S|}{|S|}.$$

Proof. For $i = 1, \dots, n$ define the following polynomial f_i in m variables

$$f_i(x) = \prod_{k \in S} (\langle u_i, x \rangle - k) = \prod_{k \in S} (u_{i1}x_1 + u_{i2}x_2 + \dots + u_{im}x_m - k).$$

Observe that for any i , $f_i(v_i) \neq 0$ since the result is a product of non-zero terms. (This follows from the fact that S is a subgroup of \mathbb{F}_p^* and thus all its members are non-zero.) Observe as well that for any $j < i$, $f_i(v_j) = 0$ since we know that $\langle u_i, v_j \rangle \in S$ and thus the result is a product which contains a zero term. By Fact 4.11 these functions are independent in \mathbb{F}_p^m . Each f_i is a polynomial of degree exactly $|S|$ in m variables, and thus a small vector space in which they reside is the span of the monomials $x_1^{i_1} x_2^{i_2} \dots x_m^{i_m}$ such that $i_1 + i_2 + \dots + i_m \leq |S|$. By the above-mentioned theorem from basic combinatorics, we know that the number of such

monomials is $\binom{m+s-1}{s}$. Using this, we see that the number of monomials required to span the subspace is

$$\sum_{i=0}^{|S|} \binom{m+i-1}{i} = \binom{m+|S|}{|S|},$$

with the equality justified as it is a known property of the binomial [17, Section 1.2.6.E]. \square

We can estimate the above binomial as follows

$$\binom{m+|S|}{|S|} = \frac{(m+|S|)!}{|S|!m!} = \frac{(m+1)(m+2)\dots(m+|S|)}{|S|!} \leq (m+|S|)^{|S|} \approx m^{|S|}.$$

This implies via Theorem 4.12 that $n \leq (m+|S|)^{|S|} \approx m^{|S|}$.

We similarly estimate the parameters in Lemma 4.10. We have that

$$m = \binom{z-1+(p-1)/|S|}{(p-1)/|S|} \approx \binom{z+p/|S|}{p/|S|} \leq (z+p/|S|)^{p/|S|} \approx z^{\frac{p}{|S|}}.$$

Further, we have that

$$n = \binom{z}{p-1} \approx z^p.$$

Thus we see that in the construction given in Lemma 4.10 that $n \approx m^{|S|}$, which we know to be approximately optimal in how large we can make m in terms of n .

4.2.3 Algebraically Nice Sets

We now show a method for constructing algebraically nice sets.

Lemma 4.13. *Let $p = 2^t - 1$ be a Mersenne prime. Then the set $S = \{2^0, 2^1, \dots, 2^{t-1}\} \subseteq \mathbb{F}_p^*$ is 3-algebraically nice. Further, the set S_0 can be chosen such that $|S_0| \geq \lceil p/2 \rceil$.*

Proof. By Euler's Theorem, which states that for any group G and element $x \in G$, $x^{|G|} = 1$, we see that, for all elements $x \in \mathbb{F}_{2^t}^*$, $x^{|\mathbb{F}_{2^t}^*|} = x^{2^t-1} = 1$. Thus every element of $\mathbb{F}_{2^t}^*$ is a root of the polynomial $x^p - 1$. Further, every finite multiplicative subgroup \mathbb{F}_r^* is cyclic, and thus it has a generator element g , meaning that every element in the group is equal to some power g^γ of g . Let g be a generator of the multiplicative subgroup $\mathbb{F}_{2^t}^*$, and fix some γ such that $1 + g + g^\gamma = 0$. Let $S_1 = \{0, 1, \gamma\}$.

For every subset $T \subseteq \mathbb{F}_p$ associate with it a unique polynomial in the ring $\mathbb{F}_2[x]/(x^p - 1)$ of the form

$$\phi_T(x) = \sum_{t \in T} x^t.$$

Note for all sets $S_1 \subseteq \mathbb{F}_p$ and all $\alpha, \beta \in \mathbb{F}_p$ with $\beta \neq 0$ that

$$\phi_{\alpha+\beta S_1}(x) = x^\alpha \phi_{S_1}(x^\beta).$$

Fix some $\alpha \in \mathbb{F}_p$ and $\beta \in S$. We will show that there exists some set S_0 such that $|S_0 \cap (\alpha + \beta S_1)| \equiv 0 \pmod{2}$, and thus which satisfies condition 3 of algebraic niceness. The polynomials $\phi_T(x)$ have degree $\leq p$ and can be viewed as incidence vectors in \mathbb{F}_2^p over the terms x^t . Define

the polynomial $\tau(x) = \gcd(x^p - 1, \phi_{S_1}(x))$. The degree of $\tau(x) \geq 1$ since g is a root of both $x^p - 1$ and of $\phi_{S_1}(x) = 1 + x + x^\gamma$. Let L be the ideal $(\tau(x))$, i.e. the linear space of multiples of $\tau(x)$. Note that $\dim(L) = p - \deg(\tau(x))$, and thus that $\dim(L) < p$ since $\deg(\tau(x)) \geq 1$. In particular, this means that $\dim(L^\perp) \geq 1$, where L^\perp is the subspace perpendicular to L , i.e. $L^\perp = \{a \mid \forall b \in L, \langle a, b \rangle = 0\}$. Further, note that

$$\phi_{\alpha+\beta S_1}(x) = x^\alpha \phi_{S_1}(x^\beta) = x^\alpha (\phi_{S_1}(x))^\beta = x^\alpha (\phi_{S_1}(x))^{\beta-1} (\phi_{S_1}(x)).$$

The second identity follows from the facts that since $\beta \in S$ it is a power of 2 and that for a polynomial f over \mathbb{F}_2 , $f(x^{2^i}) = f(x)^{2^i}$. Thus we see that $\phi_{\alpha+\beta S_1}(x) \in L$ as it is a multiple of $\phi_{S_1}(x)$ and thus also of $\tau(x)$. Finally note that any $\phi_T \in L^\perp$ yields a set T which can serve as our set S_0 , as for such a set $|S_0 \cap (\alpha + \beta S_1)|$ must be even.

To see that we can find an S_0 such that $|S_0| \geq \lceil p/2 \rceil$, note that the space L is closed under cyclic shifts, as a cyclic shift of a polynomial in L corresponds to multiplying the polynomial by x (and thus shifting the coefficients of each term over by one position) and L is defined as the set of all multiples of $\tau(x)$. This implies that L^\perp is also closed under cyclic shifts. As L^\perp is nonempty, for each coordinate i there exists some vector $\phi \in L^\perp$ with a 1 at position i . Thus if we take a random vector V from L^\perp we see that the value of each coordinate has expected weight $1/2$. Define for all $i \in [p]$ the random variable X_i to be the weight of position i in such a V . Further, define Y to be the Hamming weight of V . Then we see that

$$\mathbb{E}[Y] = \mathbb{E}[X_1 + X_2 + \dots + X_n] = \mathbb{E}[X_1] + \dots + \mathbb{E}[X_n] = p/2.$$

This implies by the pigeonhole principle of expectation [12, Section 17.2] that some vector $\phi_T \in L^\perp$ has Hamming weight at least $\lceil p/2 \rceil$, and we can use this set T as our S_0 to get one with size at least $\lceil p/2 \rceil$. \square

The largest known Mersenne prime is currently $2^{32582657} - 1$ and was discovered on September 4, 2006, by Steven Boone and Curtis Cooper [6]. We thus complete the proof of Theorem 4.2 as follows. Set $t = 32582657$ and $p = 2^t - 1$. By Lemma 4.10 we know for any odd prime p , integer $z \geq p - 1$, and subgroup $S \subseteq \mathbb{F}_p^*$ that S is $(m, n) = \left(\binom{z-1+(p-1)/|S|}{(p-1)/|S|}, \binom{z}{p-1}\right)$ combinatorially nice. Further, by Lemma 4.13 we know that for any Mersenne prime $p = 2^t - 1$ that the set $S = \{2^0, 2^1, \dots, 2^{t-1}\} \subseteq \mathbb{F}_p^*$ is 3-algebraically nice and that we can find an S_0 such that $|S_0| \geq \lceil p/2 \rceil$. Putting this together with Theorem 4.7 we see that this implies the existence of a $(3, n, p^m, p^m/2, p^m/2)$ -regular intersecting family. This in turn, by Theorem 4.4, implies the existence of a $(3, \delta, 1/2 - 3\delta p^m / (p^m/2)) = (3, \delta, 1/2 - 6\delta)$ -LDC which encodes $n = \binom{z}{p-1}$ bits to p^m bits. Choose z such that it is the smallest integer such that $n \leq \binom{z}{p-1}$ and let $n' = \binom{z}{p-1}$; provided that n is large enough this can be done such that $n' \leq 2n$ and thus we don't significantly increase the codelength. Thus given an n -bit message x , we can pad x with zeros until its length is n' and then we are able to encode it in p^m bits. If we treat p and t as constants we can approximate n by $\Theta(z^{p-1})$ and m by $\Theta(z^{(p-1)/t})$, showing that $m = \Theta(n^{1/t})$. Thus we see that the encoding is of length $p^m = 2^{O(n^{1/32582657})}$ and we are done.

Further, it has been conjectured that there are infinitely many Mersenne primes. In this case, we see have the following theorem.

Theorem 4.14. *Assuming that there are infinite Mersenne primes, for infinitely many n there exist $(3, \delta, 1/2 - 6\delta)$ -LDCs of length $2^{n^{O(\frac{1}{\log \log n})}}$.*

Proof. Following the analysis above, given some Mersenne prime p set $z = 2^p$. Then we have a $(3, \delta, 1/2 - 6\delta)$ -LDC encoding $n = \binom{2^p}{p-1} = z^{\Theta(\log z)}$ bits to $p^m = p^{\binom{z-1+(p-1)/t}{(p-1)/t}} = 2^{O(z^{\frac{\log z}{\log \log z}})}$

bits. As $\log \log n = \Theta(\log \log z)$, we see that this is equal to $2^{n^{O(\frac{1}{\log \log n})}}$ bits. \square

5 Lower Bounds

The above discussions covered various LDC and PIR constructions, and thus analysis of them gave rise to upper bounds on LDC code length. In this section we survey some lower bounds on LDC code length.

5.1 An Information-Theoretic Lower Bound

The first lower bound result for LDCs was published in 2000 by Katz and Trevisan [13]. Their result states that no LDC can exist which has a constant number of queries q and a constant information rate. The proof is information-theoretic in nature, and as the results are shown for smooth codes and not LDCs directly it relies on the translations between LDCs and smooth codes presented in Section 2.2. The general argument is as follows. Given a q -query smooth code $C : \{0, 1\}^n \rightarrow \Sigma^m$, we show how to find a subset of $O(m^{(q-1)/q})$ entries of the codeword such that these entries still contain a linear amount of information about the initial message. Then we employ information theory to show that $(\log |\Sigma|)m^{(q-1)/q} = \Omega(n)$ and thus that $m = \Omega((n/\log |\Sigma|)^{q/(q-1)})$.

We first state and prove a simple information-theoretic lemma which is used to derive the bound.

Lemma 5.1 ([13], Theorem 2). *Let $C : \{0, 1\}^n \rightarrow \{0, 1\}^m$ be a function. Assume there is an algorithm such that for all $i \in [n]$ the algorithm can recover bit x_i from $C(x)$ with probability at least $1/2 + \epsilon$ (with the probability being taken over the coin flips of the algorithm as well as uniformly over all strings x). Then*

$$n \leq \frac{m}{1 - \mathcal{H}(1/2 + \epsilon)}$$

Proof. From the definition of mutual information:

$$I(X; C(X)) \leq \mathcal{H}(C(X)) \leq m.$$

However we also have:

$$\begin{aligned} I(X; C(X)) &= \mathcal{H}(X) - \mathcal{H}(X|C(X)) \\ &\geq \mathcal{H}(X) - \sum_i \mathcal{H}(X_i|C(X)) \\ &\geq (1 - \mathcal{H}(1/2 + \epsilon))n. \end{aligned}$$

The first inequality is justified by the chain rule of relative entropy. The second inequality is justified by Fano's inequality [7, Theorem 2.11.1], which implies that if a decoder can predict X_i with probability p from knowing $C(X)$ then $\mathcal{H}(p) \geq \mathcal{H}(X_i|C(X))$. Putting this together gives the lemma. \square

Before presenting their general result, Katz and Trevisan first analyze the special case where $q = 1$ for which they produce a better bound. It is useful to walk through its proof, as it gives the intuition behind the main result. In what follows, we use the notation \Pr_x to denote that the probability is being taken uniformly over all random strings x in addition to whatever other randomness is obvious from context.

Theorem 5.2 ([13], Theorem 3). *Let $C : \{0, 1\}^n \rightarrow \Sigma^m$ be a $(1, \delta, \epsilon)$ -locally decodable code. Then*

$$n \leq \frac{\log |\Sigma|}{\delta(1 - \mathcal{H}(1/2 + \epsilon))}.$$

Proof. Let $A(\cdot, \cdot)$ be the probabilistic decoding procedure for C . For $i \in [n]$ and $j \in [m]$, define j to be ϵ -good for i if

$$\Pr_x[A(C(x), i) = x_i \mid A \text{ reads } j] \geq 1/2 + \epsilon.$$

Fix some $i \in [n]$. From the definition of an LDC we have

$$\begin{aligned} \sum_{j \in [m]} \Pr_x[A(C(x), i) = x_i \mid A \text{ reads } j] \Pr[A(\cdot, i) \text{ reads } j] \\ = \Pr[A(C(x), i) = x_i] \\ \geq 1/2 + \epsilon. \end{aligned}$$

Thus by the pigeonhole principle of expectation, we see that there must be at least one index j_1 such that $\Pr_x[A(C(x), i) = x_i \mid A \text{ reads } j_1] \geq 1/2 + \epsilon$. Thus j_1 is ϵ -good for i .

Let $C(x)_{j_1, \dots, j_k}$ denote a string which is identical to the codeword $C(x)$ except that the values at the indices j_1, \dots, j_k take on random values from the code alphabet Σ . Since LDCs are tolerant of up to δm errors, we can iterate this procedure and we see that

$$\sum_{j_2 \in [m]} \Pr_x[A(C(x)_{j_1}, i) = x_i \mid A \text{ reads } j_2] \Pr[A(\cdot, i) \text{ reads } j_2] \geq 1/2 + \epsilon,$$

where the probability is now additionally taken over all possible random values for $C(x)_{j_1}$. Further, j_2 must be different than j_1 since the value at index j_1 in the codeword which A sees is random and independent of the message string x . This again implies by the pigeonhole principle that j_2 is ϵ -good for i . We can iterate this procedure such that we obtain a set J_i of size $|J_i| = \delta m$ since the LDC is tolerant of δm errors. In this way we can create a family of such sets J_i , one for each index i , where $\forall j \in J_i, j$ is ϵ -good for i . Thus by the pigeonhole principle, there must exist some index $j' \in [m]$ which is ϵ -good for at least δn indices of the input message. We can then employ Lemma 5.1, which states that

$$\delta n \leq \frac{\log |\Sigma|}{(1 - \mathcal{H}(1/2 + \epsilon))}.$$

□

The intuition behind the previous proof was to iteratively construct sets, the existence of which implied the existence of a particular index j' which was good for a constant fraction of indices of the input message. We use the same procedure in the proof of the main result of this section. However, in this case we seek instead to iteratively create a *set* S' of indices (with $|S'| = q$) which gives a constant fraction of information about the input message and then again employ Lemma 5.1.2

Before we get to the proof of this theorem we require a pair of lemmas. For a decoding algorithm A and a set $s \subseteq [m]$, let “ A reads s ” mean that, in a particular run of A , the algorithm reads exactly the indices of the codeword which are elements of s .

Lemma 5.3 ([13], Lemma 4). *Let $C : \{0, 1\}^n \rightarrow \Sigma^m$ be a (q, δ, ϵ) -locally decodable code and let A be the associated decoding algorithm. Then for each $i \in [n]$ there exists a set M_i of disjoint tuples consisting of q or fewer elements from $[m]$ such that $|M_i| \geq \epsilon \delta m / q^2$ for all M_i . Further, for all $e \in M_i, \Pr_x[A(C(x), i) = x_i \mid A \text{ reads } e] \geq 1/2 + \epsilon/2$.*

Proof. For some set $s \subseteq [m]$ where $|s| \leq q$, we say that s is ϵ -good for i if

$$\Pr_x[A(C(x), i) = x_i \mid A \text{ reads } s] \geq 1/2 + \epsilon.$$

For all $i \in [n]$, define a hypergraph $H_i = ([m], E_i)$, with the set $[m]$ forming the set of vertices and $E_i = \{e \subseteq [m] \mid e \text{ is } \epsilon/2\text{-good for } i\}$. We say that A reads E_i if A reads some hyperedge $e \in E_i$.

A *matching* M on a hypergraph $H = (X, E)$ is a set of hyperedges such that each hyperedge is pairwise disjoint with every other. Further, a *transversal* $V \subseteq X$ for a hypergraph H is a set of vertices such that each hyperedge $e \in E$ contains at least one vertex $v \in V$. One known result from graph theory is that on a hypergraph H where the size of each hyperedge is at most q and the minimum size of a transversal is at least T , there must exist a matching of size at least T/q [4, Theorem 18.6].

Now take C , which from Theorem 2.7 we know to be a $(q, q/\delta, \epsilon)$ -smooth code, and the hypergraphs $\{H_i \mid i \in [n]\}$ defined above. Let \overline{E}_i denote the set of all hyperedges which are *not* $\epsilon/2$ -good for i . Then by the definition of a smooth code we have that

$$\begin{aligned} 1/2 + \epsilon &\leq \Pr_x[A(C(x), i) = x_i \mid A(\cdot, i) \text{ reads } E_i] \Pr[A(\cdot, i) \text{ reads } E_i] + \\ &\quad \Pr_x[A(C(x), i) = x_i \mid A(\cdot, i) \text{ reads } \overline{E}_i] \Pr[A(\cdot, i) \text{ reads } \overline{E}_i] \\ &\leq \Pr[A(\cdot, i) \text{ reads } E_i] + \\ &\quad \Pr_x[A(C(x), i) = x_i \mid A(\cdot, i) \text{ reads } \overline{E}_i] \Pr[A(\cdot, i) \text{ reads } \overline{E}_i] \\ &\leq \Pr[A(\cdot, i) \text{ reads } E_i] + (1/2 + \epsilon/2)(1 - \Pr[A(\cdot, i) \text{ reads } E_i]). \end{aligned}$$

The first inequality comes from the facts that the probability of success is at least $1/2 + \epsilon$ and that A will either read E_i or \overline{E}_i . The second step is justified by the fact that probabilities are at most 1, and the third is justified by the fact that if $A(\cdot, i)$ reads \overline{E}_i then its probability of success is less than $1/2 + \epsilon/2$. If we rearrange the terms we see that $\Pr[A(\cdot, i) \text{ reads } E_i] \geq \epsilon/(1 - \epsilon) \geq \epsilon$.

For all $e \in E_i$ let P_e be the probability that $A(\cdot, i)$ reads e . Thus by the above inequality we have that

$$\Pr[A(\cdot, i) \text{ reads } E_i] = \sum_{e \in E_i} P_e \geq \epsilon.$$

Since C is a $(q, q/\delta, \epsilon)$ -smooth code, we know that for every $j \in [m]$ that

$$\sum_{e \in E_i \mid j \in e} P_e \leq q/\delta m.$$

Now let V be a transversal for H_i . Thus $\forall e \in E_i$ we know that $e \cap V \neq \emptyset$ by the definition of a transversal. Thus

$$\sum_{e \in E_i \mid e \cap V \neq \emptyset} P_e \geq \epsilon.$$

Putting this together we see that

$$\epsilon \leq \sum_{e \in E_i \mid e \cap V \neq \emptyset} P_e \leq \sum_{j \in V} \sum_{e \in E_i \mid j \in e} P_e \leq |V|q/\delta m.$$

Hence every transversal V for H_i has $|V| \geq \epsilon \delta m/q$. By the above-mentioned result from graph theory, this implies that there exists a matching M_i on H_i with $|M_i| \geq \epsilon \delta m/q^2$. \square

This lemma is important, in that it establishes the fact (assuming without loss of generality that m divides q) that we can reduce any decoding algorithm A for an LDC to another algorithm A' with an associated partition of the codeword indices $[m]$ into a set of q -tuples such that A' queries any tuple from this partition with probability q/m . (We can pad any tuples from the above lemma which have fewer than q elements with dummy values to bring their size up to q .) Thus we see that C is also smooth over these tuples. Note that this reduces A , an algorithm with worst-case success probability of at least $1/2 + \epsilon$, to A' , an algorithm which only has an *average-case* success probability of at least $1/2 + \epsilon$ and whose worst-case success probability might be much worse. Nonetheless, this result has proven generally useful in reasoning about lower bounds for LDCs, and an example of its use will be seen in Section 5.2.

Now let $H = (X, E)$ be a hypergraph and M be a matching on H . The second lemma we will use in the proof of this section's main result gives a bound on the number of vertices we need to randomly select from H in order to have a good probability of selecting all of the vertices from a hyperedge $e \in E$.

Lemma 5.4 ([13], Lemma 5). *Let $H = (X, E)$ be a hypergraph with $|X| = m$ and for which $\forall e \in E, |e| \leq q$. Let M be a matching on H of size γm with $\gamma < 1/q$. Then there exists some $t = \Theta(\gamma^{-\frac{1}{q}} m^{\frac{q-1}{q}})$ such that for any multiset S of vertices randomly selected with replacement from H with $|S| = t$,*

$$\Pr[S \text{ contains all vertices for some hyperedge } e \in M] > 3/4.$$

Proof. The worst case is when H , and thus M , are q -uniform, i.e. when every hyperedge in H contains exactly q vertices. This is because, since $\gamma < 1/q$, we can create a new matching M' by adding vertices to all those hyperedges in M with fewer than q vertices. Thus we confine ourselves to this setting and derive a lower bound for the above probability in the general case.

Let $S = \{v_1, \dots, v_t\}$ be a multiset of vertices chosen randomly with replacement from H . Let a be a q -element subset of $[t]$. Define the random variable Y_a as follows

$$Y_a = \begin{cases} 1 & \text{if } \{v_{a_1}, \dots, v_{a_q}\} \in M \\ 0 & \text{otherwise.} \end{cases}$$

Let the random variable $Y = \sum_a Y_a$ for all q -element subsets a of $[t]$. Thus the probability in which we are interested can be rewritten as

$$\Pr[S \text{ contains all vertices for some hyperedge } e \in M] = \Pr[Y \neq 0].$$

We bound this probability by bounding its expectation and variance as follows. As elements of Y_a are chosen with replacement, all Y_a have the same distribution and so we note that

$$\mathbb{E}[Y] = \binom{t}{q} \cdot \mathbb{E}[Y_a] = \binom{t}{q} \cdot \frac{|M|}{m^q/q!} = \frac{\gamma m \binom{t}{q} q!}{m^q},$$

with the second equality coming from the fact that there are m^q different multisets of size q of elements of $[m]$ when we allow order to matter, but as it doesn't we must correct this by $q!$, the number of different permutations of q elements.

Now define another set of random variables $Z_a = Y_a - \mathbb{E}[Y_a]$. First note that $\mathbb{E}[Z_a] = \mathbb{E}[Y_a - \mathbb{E}[Y_a]] = \mathbb{E}[Y_a] - \mathbb{E}[Y_a] = 0$, by linearity of expectation. Note further that

$$\text{Var}[Y] = \mathbb{E} \left[\left(\sum_a Z_a \right)^2 \right] = \sum_a \mathbb{E}[Z_a^2] + \sum_{a, a' | a \neq a'} \mathbb{E}[Z_a Z_{a'}].$$

(Recall that a is a q -element subset of $[t]$, and that the a in the above sums ranges over all such q -element subsets of $[t]$.) We analyze the second term of the above sum as follows. Note that if $a \cap a' = \emptyset$, then Z_a and $Z_{a'}$ are independent, and so

$$\mathbb{E}[Z_a Z_{a'}] = \mathbb{E}[Z_a] \mathbb{E}[Z_{a'}] = 0.$$

If instead $a \cap a' \neq \emptyset$, then since M is a matching it cannot contain both a and a' . Thus $\mathbb{E}[Y_a Y_{a'}] = 0$ and we see that

$$\begin{aligned} \mathbb{E}[Z_a Z_{a'}] &= \mathbb{E}[(Y_a - \mathbb{E}[Y_a])(Y_{a'} - \mathbb{E}[Y_{a'}])] \\ &= \mathbb{E}[Y_a Y_{a'} - Y_a \mathbb{E}[Y_{a'}] - Y_{a'} \mathbb{E}[Y_a] + \mathbb{E}[Y_a] \mathbb{E}[Y_{a'}]] \\ &= \mathbb{E}[Y_a Y_{a'}] - \mathbb{E}[Y_a] \mathbb{E}[Y_{a'}] \\ &< \mathbb{E}[Y_a Y_{a'}] = 0. \end{aligned}$$

Thus putting this together we see that

$$\text{Var}[Y] < \frac{\gamma m \binom{t}{q} q!}{m^q}.$$

Finally, we can use the Second Moment Method (which follows from Chebyshev's Inequality - see [12, Section 22.1]) to produce a bound as follows

$$\begin{aligned} \Pr[Y = 0] &\leq \frac{\text{Var}[Y]}{\mathbb{E}^2[Y]} \\ &\leq \frac{m^q}{\gamma m \binom{t}{q} q!}. \end{aligned}$$

We wish to bound this above by $1/4$ (and thus make the probability that S hits some hyperedge in M at least $3/4$). Thus we can analyze the parameter t as follows

$$\begin{aligned} \frac{m^q}{\gamma m \binom{t}{q} q!} &= \frac{m^q (t-q)!}{\gamma m t!} \\ &= \frac{m^q}{\gamma m t (t-1) \dots (t-q+1)} \\ &\leq \frac{m^q}{\gamma m (t-q+1)^q} \\ &\approx \frac{m^q}{\gamma m t^q}. \end{aligned}$$

If we substitute $t = 4^{\frac{1}{q}} \gamma^{-\frac{1}{q}} m^{\frac{q-1}{q}} = \Theta(\gamma^{-\frac{1}{q}} m^{\frac{q-1}{q}})$ in the above equation, all variables cancel and we are left with the required $1/4$ upper bound. \square

We are now in a position to prove the main result of this section.

Theorem 5.5 ([13], Theorem 6). *Let $C : \{0, 1\}^n \rightarrow \Sigma^m$ be a (q, δ, ϵ) -locally decodable code, with A the associated decoding algorithm. Then*

$$m \geq (\epsilon \delta / q^2)^{\frac{1}{q-1}} \left(\frac{3n(1 - \mathcal{H}(1/2 + \epsilon/2))}{4 \log |\Sigma|} \right)^{\frac{q}{q-1}}.$$

Proof. From Lemma 5.3 we know that for each $i \in [n]$ there exists a set M_i of disjoint tuples of elements from $[m]$ with size $\leq q$ such that $|M_i| \geq \gamma m$ with $\gamma = \epsilon\delta/q^2$ and such that, for all $e \in M_i$, $\Pr_x[A(C(x), i) = x_i \mid A \text{ reads } e] \geq 1/2 + \epsilon/2$.

When we view each set M_i as a matching on a hypergraph with m vertices, then Lemma 5.4 shows that there exists some set S with size $|S| = t = \Theta((\epsilon\delta/q^2)^{-\frac{1}{q}} m^{\frac{q-1}{q}})$ such that for $3n/4$ indices $i_1, \dots, i_{3n/4} \in [n]$, S hits each of $M_{i_1}, \dots, M_{i_{3n/4}}$. In other words, this means that if A reads each of these t elements from $C(x)$ then this gives enough information such that each of $3n/4$ bits of x can be recovered with probability at least $\epsilon/2$. Thus by Theorem 5.2, we know that

$$t \log |\Sigma| \geq \frac{3n(1 - \mathcal{H}(1/2 + \epsilon/2))}{4}.$$

If we plug in $(\epsilon\delta/q^2)^{-\frac{1}{q}} m^{\frac{q-1}{q}}$ for t we see that

$$\begin{aligned} (\epsilon\delta/q^2)^{-\frac{1}{q}} m^{\frac{q-1}{q}} \log |\Sigma| &\geq \frac{3n(1 - \mathcal{H}(1/2 + \epsilon/2))}{4} \\ m^{\frac{q-1}{q}} &\geq (\epsilon\delta/q^2)^{\frac{1}{q}} \left(\frac{3n(1 - \mathcal{H}(1/2 + \epsilon/2))}{4 \log |\Sigma|} \right) \\ m &\geq (\epsilon\delta/q^2)^{\frac{1}{q-1}} \left(\frac{3n(1 - \mathcal{H}(1/2 + \epsilon/2))}{4 \log |\Sigma|} \right)^{\frac{q}{q-1}}, \end{aligned}$$

proving the theorem. This thus shows that $m = \Omega((n/\log |\Sigma|)^{\frac{q}{q-1}})$, where the constant depends on ϵ, δ , and q . \square

5.2 A Quantum Lower Bound

While the best known lower bounds for general LDCs are still polynomial, Kerenidis and de Wolf have shown in [15] an exponential lower bound for LDCs with $q = 2$ queries via an argument which uses tools from quantum computing. The general structure of the argument is as follows. We consider some LDC C , and we use Theorem 2.7 to reduce C to a $(2, c, \epsilon)$ -smooth code. We then use another Fact from [15] (see also the discussion after Lemma 5.3) to further reduce this to a binary $(2, c \cdot 2^\ell, \epsilon/2^{2\ell})$ -smooth code which is good on average. We then use yet another reduction to show that we can transform any decoding algorithm into one which outputs the XOR of the bits it queries. Next, we employ a reduction which transforms 2 classical queries into 1 quantum query and show that we can recover the XOR of two bits with perfect probability with 1 quantum query. We thus have a quantum random access code of $\log m$ qubits, and since a linear lower bound for such a code is known, we see that $m = 2^{\Omega(n)}$.

We begin with a brief introduction to the basics of quantum computing.

5.2.1 Quantum Notation

First some notation. Consider the vector space \mathbb{C}^2 along with the standard inner product. We choose some pair of orthonormal vectors from this space to serve as a basis - label these $|0\rangle$ and $|1\rangle$, and identify them with the vectors $\begin{pmatrix} 1 \\ 0 \end{pmatrix}$ and $\begin{pmatrix} 0 \\ 1 \end{pmatrix}$ respectively. This is an example of Dirac notation, which proves very useful for the sorts of operations we perform on complex vectors in quantum computation. In this vein, we further write $\langle\phi| = |\phi\rangle^*$ for the complex conjugate transpose of a vector $|\phi\rangle$. Finally, the inner product of two vectors ϕ and ψ is written $\langle\psi|\phi\rangle$. Thus the Euclidean norm of ϕ is $\|\phi\| = \sqrt{\langle\phi|\phi\rangle}$.

A *qubit* is a unit vector in this space, and can be written as a linear combination of the basis vectors

$$\alpha |0\rangle + \beta |1\rangle = \begin{pmatrix} \alpha \\ \beta \end{pmatrix}.$$

Thus the qubit is said to be in a *superposition* of the basis states, with each basis state contributing an *amplitude* to the qubit's state. One can think of this as the vector being in both of the basis states at the same time, with each basis state contributing a given amplitude or weight.

5.2.2 Quantum Registers

Just as a single qubit is a unit vector in the 2-dimensional complex vector space, an m -qubit system is a unit vector in the 2^m -dimensional tensor space $\mathbb{C}^2 \otimes \dots \otimes \mathbb{C}^2$. For example, a 2-qubit system evolves within the 4-dimensional vector space \mathbb{C}^4 and has basis states $|0\rangle \otimes |0\rangle$, $|0\rangle \otimes |1\rangle$, $|1\rangle \otimes |0\rangle$, and $|1\rangle \otimes |1\rangle$. We often abbreviate $|1\rangle \otimes |0\rangle$ as $|1\rangle |0\rangle$, $|10\rangle$, or even $|2\rangle$ as 10 in binary radix is 2 in decimal. Thus an m -qubit system ϕ can be in any unit-length superposition of the 2^m basis states and can be expressed in the form

$$|\phi\rangle = \sum_{i \in \{0,1\}^m} \alpha_i |i\rangle.$$

5.2.3 Quantum Measurement

There are two kinds of operations which the axioms of quantum mechanics allow us to perform on quantum states: measurement and evolution. While there is a more general form of measurement than that which will be presented here, the details aren't necessary for what we will cover and so we omit them. When we measure a quantum state, we can do so with respect to a given orthonormal basis in the complex vector space in which the state lies and we then see one of the basis states as a result. After this the quantum state collapses to whatever classical state we saw, and we lose whatever information was stored in the rest of the state. Which state we see depends on the amplitudes of the respective basis states which make up the quantum state we are measuring. More precisely, the probability that we see a given state is equal to the squared norm of that state's amplitude. From this we see that the squared norms of the amplitudes must sum to 1:

$$\sum_{i \in \{0,1\}^m} |\alpha_i|^2 = 1.$$

5.2.4 Quantum Evolution

The other thing we can do to quantum states is to apply certain transformations to them. This process is called *evolution*, and another axiom of quantum mechanics requires that all such transformations be linear. Thus we can write any such transformation as a complex matrix U , and when we apply U to the state $|\phi\rangle$ it evolves to $U|\phi\rangle$.

Since measuring the resulting state should still give us a probability distribution, the constraint that $\sum_{i \in \{0,1\}^m} |\alpha'_i|^2 = 1$ (with the α'_i being the amplitudes of the resulting state) remains. Thus we see that the transformation must preserve norms and hence is a *unitary* transformation. An equivalent definition of unitary transformations states them to be all transformations whose inverse is equal to their complex conjugate transpose - i.e. $U^{-1} = U^*$. Thus all evolutionary

transformations are invertible. Note that this is in contrast with measurements, which are by their nature destructive - once we measure a state $|\phi\rangle$ and have it collapse to some classical state $|i\rangle$, we cannot recover the original state $|\phi\rangle$.

For a more complete reference on quantum computation, we refer the reader to [21].

5.2.5 Proof of the Quantum Lower Bound

We first state two results from other works which we will use.

Fact 5.6 ([15, Lemma 2]). *Let $C : \{0, 1\}^n \rightarrow \Sigma^m$ be a $(2, c, \epsilon)$ -smooth code. Then there exists another $(2, c \cdot 2^\ell, \epsilon/2^{2^\ell})$ -smooth code C' which is good on average. Further, the decoder outputs either the XOR or its negation of the two queried bits.*

A quantum random access code is an encoding $x \mapsto \rho_x$ of n -bit strings into m -qubit states ρ_x such that any bit x_i can be recovered with some probability $p \geq 1/2 + \epsilon$ from ρ_x . The following lower bound is known on the length of such codes.

Theorem 5.7 ([20]). *An encoding $x \mapsto \rho_x$ of n -bit strings to m -qubit states with recovery probability on average over all x of at least p has $m \geq (1 - \mathcal{H}(p))n$.*

With these, we prove the following theorem.

Theorem 5.8 ([15, Theorem 4]). *If $C : \{0, 1\}^n \rightarrow \{0, 1\}^m$ is a $(2, \delta, \epsilon)$ -locally decodable code, then $m = 2^{\Omega(n)}$.*

Proof. We start with the smooth code, using the reduction discussed previously. By Fact 5.6, we see that without loss of generality we can transform this into a uniformly smooth code with the added quality that the decoder only outputs the XOR (or its negation) of the two bits which it queries.

We first formalize the concept of a query in the quantum world. There are two equivalent (up to the requirement for some additional workspace qubits) formalizations of a query to an m -bit string y . The first is as follows: for $j \in [m]$ and $b \in \{0, 1\}$ (with b being the target bit), we define a query to be a unitary transformation which maps

$$|j\rangle |b\rangle \rightarrow |j\rangle |b \oplus j\rangle.$$

Such an operation may be applied to any superposition of states. The second formalization, which is that which we will use in what follows, is instead a unitary transformation which maps

$$|c\rangle |j\rangle \rightarrow (-1)^{c y_j} |c\rangle |j\rangle.$$

In this case, c is called the control bit, and as can be seen the phase $(-1)^{y_j}$ is only added if c is 1.

We now show the reduction of 2 classical queries to 1 quantum query. By assumption, the classical decoder outputs the XOR of the two bits it queries. Consider the following state

$$\frac{1}{\sqrt{2}}(|1\rangle |1\rangle + |1\rangle |2\rangle).$$

If the decoder makes such a query, the resulting state is

$$\frac{1}{\sqrt{2}}((-1)^{a_1} |1\rangle |1\rangle + (-1)^{a_2} |1\rangle |2\rangle) = \frac{(-1)^{a_1}}{\sqrt{2}}(|1\rangle |1\rangle + (-1)^{a_1 \oplus a_2} |1\rangle |2\rangle),$$

from which we can recover $a_1 \oplus a_2$ with certainty. Thus we see that a decoder which uses one quantum query can compute the same output as a decoder which uses two classical queries.

It is immediate to see that the above reduction of 2-query uniformly smooth classical codes to 1-query uniformly smooth quantum codes induces such a random access code with $p = 1/2 + \epsilon/2^{2^\ell}$. Since we are querying a quantum state with $\log m$ qubits, we see from Theorem 5.7 that $\log m \geq (1 - \mathcal{H}(p))n$ and thus that $m = 2^{\Omega(n)}$, proving the lower bound. It should be mentioned that applying the techniques in this proof allow us to improve the bound given in Theorem 5.5. \square

5.2.6 Lower Bounds for LDCs with more than 2 Queries

We briefly state here the best-known lower bounds for general LDCs. In [15], Kerenidis and de Wolf prove the following result:

Theorem 5.9. *If $C : \{0, 1\} \rightarrow \{0, 1\}^m$ is a (q, δ, ϵ) -LDC, then*

$$m = \Omega \left(\left(\frac{n}{\log n} \right)^{1+1/(\lceil q/2 \rceil - 1)} \right),$$

where the constant depends on q , δ , and ϵ .

Recently, Woodruff improved this result slightly to $\Omega(n^{1+1/(\lceil q/2 \rceil - 1)}/\log n)$ [28].

6 Conclusion

We have surveyed the theoretical underpinnings as well as the main results in the fields of Locally Decodable Codes and Private Information Retrieval, covering historical results as well as the best known upper and lower bounds. The major open questions in this field involve the current enormous gap between the known upper and lower bounds. Yekhanin's breakthrough result is the first which points to a positive answer to this question, but it is still open. Furthermore, are sub-linear schemes for PIR and sub-exponential-length LDCs possible? Is the gap between classical PIR and Quantum PIR, with the best known 2-server classical scheme having $O(n^{1/3})$ complexity while the best known quantum scheme has $O(n^{1/32582657})$ complexity, a true separation? There is evidence that this is the case, as Razborov and Yekhanin have shown that all bilinear group-based PIR schemes have an $\Omega(n^{1/3})$ lower bound for 2-server schemes [22]. As all known schemes are based on such groups, it seems likely that this bound holds for the general 2-server classical case. Finally and more generally, what are the optimal PIR schemes and LDCs, i.e. how do we close the gap between the known bounds?

For further reading, we point the reader toward the references.

Acknowledgments

I would like to acknowledge Ronald de Wolf for taking so much time to get me started on this field. This paper would not exist but for his insights and encouragement.

References

- [1] L. Babai, L. Fortnow, L. Levin, and M. Szegedy. Checking computations in polylogarithmic time. In *Proceedings of 23rd ACM STOC*, pages 21–31, 1991.

- [2] L. Babai, L. Fortnow, N. Nisan, and A. Wigderson. BPP has subexponential time simulations unless EXPTIME has publishable proofs. *Computational Complexity*, 3(2):307–318, 1993.
- [3] A. Beimel, Y. Ishai, E. Kushilevitz, and J. Raymond. Breaking the $O(n^{1/(2k-1)})$ barrier for information-theoretic Private Information Retrieval. In *Proceedings of 43rd IEEE FOCS*, pages 261–270, 2002.
- [4] C. Berge. *Graphs and Hypergraphs*. North-Holland, Amsterdam, 1973.
- [5] B. Chor, O. Goldreich, E. Kushilevitz, and M. Sudan. Private information retrieval. *Journal of the ACM*, 45(6):965–981, 1998. Earlier version in FOCS’95.
- [6] C. Cooper and S. Boone. <http://www.mersenne.org/32582657.htm>.
- [7] T. M. Cover and J. A. Thomas. *Elements of Information Theory*. Wiley, 1991.
- [8] P. Gemmell, R. Lipton, R. Rubinfeld, M. Sudan, and A. Wigderson. Self-testing/correcting for polynomials and for approximate functions. In *Proceedings of 23rd ACM STOC*, pages 32–42, 1991.
- [9] Y. Gertner, Y. Ishai, E. Kushilevitz, and T. Malkin. Protecting data privacy in private information retrieval schemes. *Journal of Computer and Systems Sciences*, 60(3):592–629, 2000. Earlier version in STOC 98.
- [10] O. Goldreich, H. Karloff, L. Schulman, and L. Trevisan. Lower bounds for linear locally decodable codes and private information retrieval. *Computational Complexity*, 15(3):263–296, 2006. Earlier version in Complexity’02. Also on ECCC.
- [11] R. Hamming. Error detecting and error correcting codes. *Bell System Technical Journal*, 29:147–160, 1950.
- [12] S. Jukna. *Extremal Combinatorics*. EATCS Series. Springer, 2001.
- [13] J. Katz and L. Trevisan. On the efficiency of local decoding procedures for error-correcting codes. In *Proceedings of 32nd ACM STOC*, pages 80–86, 2000.
- [14] K. Kedlaya and S. Yekhanin. Locally decodable codes from nice subsets of finite fields and prime factors of mersenne numbers. Technical report, ECCC TR–07–040, 2007. Available at <http://www.eccc.uni-trier.de/eccc/>.
- [15] I. Kerenidis and R. de Wolf. Exponential lower bound for 2-query locally decodable codes via a quantum argument. *Journal of Computer and Systems Sciences*, 69(3):395–420, 2004. Earlier version in STOC’03. quant-ph/0208062.
- [16] I. Kerenidis and R. de Wolf. Quantum symmetrically-private information retrieval. *Information Processing Letters*, 90(3):109–114, 2004. quant-ph/0307076.
- [17] D. E. Knuth. *The Art of Computer Programming. Volume 1: Fundamental Algorithms*. Addison-Wesley, third edition, 1997.
- [18] E. Kushilevitz and R. Ostrovsky. Single-database computationally private information retrieval. In *Proceedings of 38th IEEE FOCS*, pages 364–373, 1997.
- [19] C. Lu, O. Reingold, S. Vadhan, and A. Wigderson. Extractors: Optimal up to constant factors. In *Proceedings of 35th ACM STOC*, pages 602–611, 2003.

- [20] A. Nayak. Optimal lower bounds for quantum automata and random access codes. In *Proceedings of 40th IEEE FOCS*, pages 369–376, 1999. quant-ph/9904093.
- [21] M. A. Nielsen and I. L. Chuang. *Quantum Computation and Quantum Information*. Cambridge University Press, 2000.
- [22] A. Razborov and S. Yekhanin. An $\Omega(n^{1/3})$ lower bound for bilinear group based private information retrieval. In *Proceedings of 47th IEEE FOCS*, pages 739–748, 2006.
- [23] C. E. Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27:379–423, 623–656, 1948.
- [24] C. E. Shannon. Communication theory of secrecy systems. *Bell System Technical Journal*, 28:656–715, 1949.
- [25] M. Sudan. Coding theory: Tutorial and survey. 2001.
- [26] M. Sudan, L. Trevisan, and S. Vadhan. Pseudorandom generators without the XOR lemma. In *Proceedings of 31st ACM STOC*, pages 537–546, 1999.
- [27] R. de Wolf. Personal communication, November 2006.
- [28] D. Woodruff. New lower bounds for general locally decodable codes. Technical report, ECCC TR-07-006, 2007. Available at <http://www.eccc.uni-trier.de/eccc/>.
- [29] S. Yekhanin. New locally decodable codes and private information retrieval schemes. Technical report, 2006. Available at ECCC TR06-127. Conference version to appear in STOC’07.