On the Efficiency of Local Decoding Procedures for Error-Correcting Codes

Jonathan Katz* Luca Trevisan†

ABSTRACT

We consider error-correcting codes where a bit of the message can be probabilistically recovered by looking at a limited number of bits (or blocks of bits) of a (possibly) corrupted encoding. Such codes can be derived from multivariate polynomial encodings, and have several applications in complexity theory, such as worst-case to average-case reductions, probabilistically checkable proofs, and private information retrieval.

Such codes could have practical applications if they had at the same time constant information rate, the ability to correct a linear number of errors, and very efficient (ideally, constant-time) reconstruction procedures. In particular they would give fault-tolerant data storage with unlimited scalability.

We show a negative result on the existence of such codes; namely, that linear encoding length is incompatible with a decoding procedure making a constant number of queries (which is necessary if one is to have constant reconstruction time). In particular, if a bit of a message of length n can be retrieved by looking at q blocks of length l, and the reconstruction procedure is robust to a fraction δ of errors, then the encoding is made of $m = \Omega(\text{poly}(1/q, \delta, \epsilon)(n/l)^{q/(q-1)})$ blocks of length l.

This is the first lower bound for this class of codes. Our bound is far from the known (exponential) upper bound when q is a constant. Closing this gap remains a challenge.

1. INTRODUCTION

Error-correcting codes are typically used to reliably transmit information over noisy channels. An equally useful application of error-correcting codes is to reliably store information on a medium whose content may be partially corrupted over time (or whose reading device is subject to errors). For example, the data stored in music CDs and in CD-ROMs is encoded using Reed-Solomon codes. In applications to data storage (and also in applications to data transmission) a message is typically divided into small blocks, and then each block is encoded separately. This allows efficient retrieval of the information, since one need decode only the portion of data one is interested in; on the other hand, it limits reliability, since even if the data in one block (out of possibly millions of others) is corrupted, then some information is irreparably lost. One would have much higher robustness if the entire information were encoded as a *single* codeword in an error-correcting code. One could then corrupt even a large fraction of the entire encoding and still not lose any information. The downside is that even using codes that are decodable in linear time, one still has to read the entire encoding in order to retrieve even a single bit of its original content. More radically, one can think of encoding an entire library as a single codeword (this example is used in [12]), and then splitting the encoding into several disks. Then the whole content of the library would be resilient to catastrophic losses. Once again, if reconstructing a small piece of the original data involves processing the entire encoding, this is not an implementable proposal.

Locally decodable codes.

During the course of investigations on the applications of error-correcting codes to complexity theory and cryptography, it has been proved that certain codes admit sub-linear time randomized decoding procedures. In particular, one can reconstruct a single bit of the original data by reading only a small number of randomly chosen locations in the encoding. Codes based on multivariate polynomials have this property, and this property, along with the reconstruction procedure, has been described in the language of self-correcting computations and random self-reducibility (see e.g. [6; 13; 9; 8; 10]). A more explicit observation that multivariate polynomials yield error-correcting codes with efficient reconstruction can be found in works on probabilistically checkable proofs; in particular, an explicit statement can be found in [4]. Recent work on worst-case to average-case reductions with applications to pseudorandomness [5] (see also [16] for explicit statements in terms of error-correction), as well as work on private information retrieval [7] also focuses, more or less explicitly, on codes having efficient reconstruction procedures.

*`jkatz@cs.columbia.edu`. Department of Computer Science, Columbia University. Work supported in part by a DoD NDSEG Fellowship.

†`luca@cs.columbia.edu`. Department of Computer Science, Columbia University.

Efficiency.

There are two main parameters measuring the performance of an error-correcting code. For a code $\mathbf{C} : \{0,1\}^n \to \{0,1\}^m$ mapping an n-bit input into an m-bit encoding, the *information rate* is defined as n/m, and is the "amortized" amount of information stored in every bit of the encoding. One would like this parameter to be as large possible—ideally, lower bounded by a fixed constant for every n. The other parameter is the fraction of the m bits of the encoding that can be corrupted while still allowing correct reconstruction. One would like this parameter to be as large as possible as well—ideally, lower bounded by a fixed constant. In the context of efficient randomized reconstruction, two additional efficiency parameters are the running time of the reconstruction procedure (for reconstructing one bit) and the error probability. The ideal error-correcting code would have constant rate, resilience to a linear number of errors in the encoding, and a reconstruction procedure having a (small) constant error probability and running in constant time. Such a code would implement fault-tolerant data storage with optimal scalability. By doubling the size of the storage support one could store twice as much data, and be able to correct twice as many errors, while random access time to the original data (as well as the probability of incorrect decoding, even in the presence of errors) would remain the same. In this paper we show that such a code cannot exist. Note that, in contrast, there are error correcting codes having constant rate, correcting a linear number of errors, and having a reconstruction procedure that reconstructs all the original data in linear time, which has constant *amortized* time per bit of the original message.

Our Results.

The precise statement of our result is as follows. Suppose there is an encoding of n bits of information into m blocks each of length l, and that there is a reconstruction procedure that can decode a single bit of a message with probability $1/2+\epsilon$ when given random access to an input that is within distance δm from a correct encoding; also assume that the reconstruction procedure reads at most q bits of its input ($q \geq 2$). Then $m = \Omega((n/l)^{q/(q-1)})$ where the constant in the Ω notation depends on ϵ, δ and q. For the special case $q = 1$, we show that n is bounded by a constant of size $O(l\epsilon^{-O(1)}\delta^{-1})$ (which is independent of m). Note that in proving these lower bounds we do not assume that the reconstruction procedure is efficiently computable, but only assume that it reads a bounded number of (blocks of) bits from its input.

Our result shows that constant-time reconstruction procedures are impossible for codes having constant rate. In fact, every reconstruction procedure for a code having constant rate must read $\Omega(\log n/\log\log n)$ bits of the encoding.

Comparison with previous results.

This is the first lower bound proved for this class of error-correcting codes, and the first time that an *efficiency* restriction in the reconstruction procedure (although a very strong one, and information-theoretic in nature) implies a limitation to the *information rate* of a code.

Our bounds are far from known achievable upper bounds. In particular, for $l = 1$, constant q, and sufficiently small (but constant) δ and ϵ, the best known codes allowing reconstruction by reading q bits use encodings of length $2^{O(n^{1/(q-1)})}$.[1] Encodings using multivariate low-degree polynomials concatenated with standard error-correcting codes give codes with $m = n^{1+o(1)}$ and the reconstruction procedure reads $\operatorname{poly}\log n$ bits and runs in $\operatorname{poly}\log n$ time [4]. It remains an open question whether one can achieve polynomial length in the encoding while having a reconstruction procedure with constant query complexity, or linear length with a polylogarithmic number of queries.

There are better positive results for codes having local *checking* procedures. An efficient checker for a code is a randomized procedure that looks at a small number of entries of a given string, and accepts with probability 1 if the given string is a valid codeword, and accepts with probability, say, less than $3/4$ if the given string is far from every codeword. Results about "low-degree tests" as developed for PCP constructions [3; 2; 15] imply the existence of codes having an encoding of polynomial length and a checking procedure that only reads a constant number of entries. We do not know how to prove negative results on the trade-off between query complexity and encoding length for locally checkable codes, and we point it out as an interesting open question.

Techniques.

The known constructions of locally checkable codes, be they linear-algebraic [7] or based on low-degree polynomials [6], have the property that each individual query is uniformly distributed over the bits of the input string. One can show that, for certain ranges of the parameters, this already guarantees error-correction, since if the fraction δ of errors is small compared to $1/q$ (where q is the number of queries), then a union bound tells us that there is a high probability (at least $1-\delta q$) of reading q non-corrupted entries. The analysis in [10] allows treatment of values of δ that are bigger than $1/q$, but the reconstruction procedure still has the property of making uniformly distributed queries.

One can conceive of codes having a reconstruction procedure that queries with very high probability a certain particular entry; in fact, one can modify any construction so that it makes a certain query with probability 1 (and then ignores it). Our first result is that this is more-or-less the only situation that can occur. If a reconstruction procedure makes certain queries with probability much higher than under the uniform distribution, then it can work more-or-less equally well without making those queries. Intuitively, the reason is that no matter how an adversary corrupts a fraction δ of the entries of a codeword, the reconstruction is still guaranteed to work. If some entries are queried with very high probability (say, more than $q/\delta m$) and the procedure is making decisive use of them, then the adversary has an easy time corrupting those entries and affecting the performance of the decoding procedure. We call a reconstruction procedure *smooth* if the distribution of queries is such that no entry of the encoding is ever queried by the reconstruction procedure with probability much higher than in the uniform distribution (see Section 2 below for a precise quantitative definition). We prove that every code having a good reconstruction procedure also has a smooth decoding procedure.

[1] No published construction gives quite this bound, but a variation of [1] due to Goldreich [11] does. Multivariate polynomial encodings, concatenated with a binary error correcting code, do give, without further elaborations, locally-decodable codes of length $2^{n^{O(1/q)}}$.

This reduction is important because our lower bound uses smoothness in an essential way.

In order to prove our lower bound, we show that for every smooth encoding $\mathbf{C} : \{0,1\}^n \to (\{0,1\}^l)^m$ that has a q-query reconstruction procedure, it is possible to find a subset of $O(m^{(q-1)/q})$ entries of the encoding such that $\mathbf{C}$, restricted to such entries, still encodes a linear amount of information about its input. An information-theoretic argument then implies that $lm^{(q-1)/q} = \Omega(n)$, which is our main result. This approach was introduced in [14] to prove a lower bound for private information retrieval. Implementing this idea for smooth encodings requires some new technical work.

The main intermediate step in our analysis can be described as follows. Let $\mathbf{C} : \{0,1\}^n \to (\{0,1\}^l)^m$ be a smooth encoding that has a q-query reconstruction procedure. For every index i in $\{1,\dots,n\}$ and q-tuple $(j_1,\dots,j_q)$ of indices in $\{1,\dots,m\}$, we say that $(j_1,\dots,j_q)$ is *good* for i if the encoding $\mathbf{C}(x)$, restricted to the entries $(j_1,\dots,j_q)$, allows prediction of the value of x_i with probability noticeably bounded away from $1/2$ (the probability being taken over a random choice of x in $\{0,1\}^n$). We show that for every i, there are $\Omega(m)$ *disjoint* q-tuples that are good for i. Let S be a random subset of $\{1,\dots,m\}$ of size $\Theta(m^{(q-1)/q})$. We show that on average (over the choice of S) there are $\Omega(n)$ indices i in $\{1,\dots,n\}$ such that there is a q-tuple in S which is good for i. In particular, there *is* a set S^* with this property. If we restrict $\mathbf{C}(x)$ to the entries in S^*, we see that we maintain $\Omega(n)$ bits of information about x.

Organization of the paper.

In Section 2 we give definitions of locally decodable codes and of codes admitting smooth decodings, and we prove a reduction between the two notions. Our negative results are proved in Section 3. Section 4 discusses extensions of our results as well as some open questions.

2. PRELIMINARIES AND DEFINITIONS

Throughout this paper we use the following notation: $[m] \equiv \{1,2,\dots,m\}$, the metric $d(\cdot,\cdot)$ always refers to the Hamming distance between two codewords, and the function $\mathcal{H}(\cdot)$ is the standard entropy function.

We now present the formal definition of a locally decodable code. The reader will notice that the definitions introduced in this section allow for both adaptive and non-adaptive probabilistic decoding procedures. ¿From here on, however, we restrict ourselves to the non-adaptive case unless explicitly stated otherwise.

Definition 1 For fixed δ, ϵ, and integer q we say that $\mathbf{C} : \{0,1\}^n \to \Sigma^m$ is a (q,δ,ϵ)-*locally decodable code* if there exists a probabilistic algorithm A such that:

- For every $x \in \{0,1\}^n$, for every $y \in \Sigma^m$ with $d(y,\mathbf{C}(x)) \le \delta m$, and for every $i \in [n]$, we have:

  $$\Pr[A(y,i) = x_i] \ge 1/2 + \epsilon,$$

  where the probability is taken over the internal coin tosses of A.

- In every invocation, A reads at most q indices of y (in fact, without loss of generality, we can assume that A reads exactly q indices of y).

An algorithm A satisfying the above requirements is called a (q,δ,ϵ)-*local decoding algorithm* for $\mathbf{C}$. □

All known constructions of local decoding algorithms [6; 10; 7; 16] have the additional feature that indices of the codeword are probed uniformly at random. While one can construct arbitrary examples in which this is no longer true (for example, an algorithm which always reads a particular index and ignores the value), it seems natural that those indices which are read very often cannot give much information about the original data. Indeed, in this case, corruption of the bits which are read very often would ruin the success probability of the decoding algorithm. This idea motivates our definition of *smooth* codes.

Definition 2 For fixed c, ϵ, and integer q we say that $\mathbf{C} : \{0,1\}^n \to \Sigma^m$ is a (q,c,ϵ)-*smooth code* if there exists a probabilistic algorithm A such that:

- For every $x \in \{0,1\}^n$ and for every $i \in [n]$, we have:

  $$\Pr[A(\mathbf{C}(x),i) = x_i] \ge 1/2 + \epsilon.$$

- For every $i \in [n]$ and $j \in [m]$, we have:

  $$\Pr[A(\cdot,i) \text{ reads index } j] \le c/m.$$

- In every invocation, A reads at most q indices of y.

(The probabilities are taken over the internal coin tosses of A.) An algorithm A satisfying the above requirements is called a (q,c,ϵ)-*smooth decoding algorithm* for $\mathbf{C}$. □

Note that unlike a locally decodable code, the decoding procedure for a smooth code is required to work only for valid codewords of $\mathbf{C}$; it is not required to work for all strings within some radius of valid codewords. Instead, we only require that A not read any particular index "too often" (in a way made formal by the definition). This is what gives rise to the name *smooth*: the distribution of indices which are read is "smooth" as opposed to being peaked at some particular index or set of indices. Note also that in the case of smooth codes we do not require that A read *exactly* q indices of y. In fact, it may be advantageous for an algorithm to sometimes read fewer than q indices to maintain the smoothness requirement.

As mentioned previously, all known examples of locally decodable codes are smooth (in fact, in known constructions the queries are uniformly distributed; i.e., the codes are (q,q,ϵ)-smooth). The following theorem shows that this is always the case.

THEOREM 1. *Let $\mathbf{C} : \{0,1\}^n \to \Sigma^m$ be a (q,δ,ϵ)-locally decodable code. Then $\mathbf{C}$ is also a $(q,q/\delta,\epsilon)$-smooth code.*

PROOF. Let algorithm A be a (q,δ,ϵ)-local decoding algorithm for $\mathbf{C}$. For all $i \in [n]$, let S_i be the set of indices $j \in [m]$ such that $\Pr[A(\cdot,i) \text{ reads index } j] > q/\delta m$. Since A reads at most q indices in every invocation, it is clear that, for all i, the number of indices in S_i can be at most δm.

Define algorithm A' as follows: $A'(\cdot,i)$ runs $A(\cdot,i)$ in a black-box manner by reading indices from the codeword, as requested by A, and returning their values to A. The only exception is that if A requests to read an index in S_i, A' does not read that index, but instead simply returns 0 to A. We now have:

$$\Pr\left[A'(\mathbf{C}(x),i) = x_i\right] = \Pr\left[A(\mathbf{C}'(x),i) = x_i\right], \qquad (1)$$

where:

$$\mathbf{C}'(x)_j = \begin{cases} 0 & \text{if } j \in S_i \\ \mathbf{C}(x)_j & \text{otherwise} \end{cases}.$$

Since the size of S_i is at most δm, we have $d(\mathbf{C}(x), \mathbf{C}'(x)) \leq \delta m$. Because A is a (q, δ, ϵ)-local decoding algorithm, the probability on the right side of (1) is at least $1/2 + \epsilon$. A' is therefore a $(q, q/\delta, \epsilon)$-smooth decoding algorithm for $\mathbf{C}$, and the theorem follows. $\square$

3. MAIN RESULTS

We now turn to proving lower bounds on the length of locally decodable codes.

We begin with a theorem in Section 3.1 which relates certain information-theoretic concepts to lower bounds for encoding length. We will appeal to this result often in the sections which follow. In Section 3.2 we consider local decoding algorithms which read only one index of the codeword (i.e., $q = 1$). In this case, we work directly with the definition of $\mathbf{C}$ as a locally decodable code and show that such codes cannot exist when the length of the input string is too large (see below for an exact statement of results). In Section 3.3 we turn to the case of $q > 1$. Here, we find it more convenient to view $\mathbf{C}$ as a smooth code. We show a super-linear lower bound for the length of an encoding in this case.

It is interesting to note that an application of the approach from Section 3.3 to the case $q = 1$ also gives an absolute upper bound on the length of the input string. However, the bound is not as tight as that given in Section 3.2 using a different approach.

3.1 Information Theory

Our bounds on locally decodable codes are all obtained via information-theoretic arguments. Therefore, although the following result is a straightforward application of information theory, we single it out because we will refer to it often in the remainder of the paper.

THEOREM 2. *Let $\mathbf{C} : \{0,1\}^n \to R$ be a function. Assume there is an algorithm A such that:*

$$\forall i \in [n], \Pr_x [A(\mathbf{C}(x), i) = x_i] \geq 1/2 + \epsilon.$$

(Where the notation $\Pr_x[\ ]$ indicates that the probability is taken over the random coins of A as well as over all strings x.) Then $\log |R| \geq (1 - \mathcal{H}(1/2 + \epsilon))n$.

PROOF. Let $I(x; \mathbf{C}(x))$ be the mutual information between x and $\mathbf{C}(x)$. Then:

$$I(x; \mathbf{C}(x)) \leq \mathcal{H}(\mathbf{C}(x)) \leq \log |R|.$$

But we also have:

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

Combining these results gives the stated theorem. $\square$

3.2 Impossibility Result for $q = 1$

We begin with a particularly strong result for the case $q = 1$: fix ϵ, δ, and encoding alphabet Σ. Then there is a constant upper bound on the length of the data which can be encoded by a $(1, \delta, \epsilon)$-locally decodable code. The result is independent of the code length—even when the codeword is exponentially longer than the input, such codes cannot be constructed. The intuition is as follows: given such a code, there must be some index of the codewords which contains information about a constant fraction of the bits of the original data. But application of Theorem 2 shows that this cannot occur when the input is too long. Stated another way, the Theorem shows that there exist no families of $(1, \delta, \epsilon)$-locally decodable codes.

THEOREM 3. *Let $\mathbf{C} : \{0,1\}^n \to \Sigma^m$ be a $(1, \delta, \epsilon)$-locally decodable code. Then:*

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

PROOF. For $i \in [n]$, $j \in [m]$, say that j is ϵ-*good* for i if:

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

Fix $i \in [n]$. By definition of a locally decodable code, we have:

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

But then there must exist some index j_1 such that:

$$\Pr_x [A(\mathbf{C}(x), i) = x_i \mid A(\cdot, i) \text{ reads } j_1] \geq 1/2 + \epsilon.$$

In other words, j_1 is good for i.

Denote by $\Delta_{j_1, j_2, \ldots} \in \Sigma^m$ a "perturbation" vector which is zero in all positions except $j_1, j_2, \ldots$, where it takes on value(s) randomly chosen from Σ. Vector $\mathbf{C}(x) \oplus \Delta_{j_1, j_2, \ldots}$ (where $\oplus$ denotes addition in the appropriate field) is then identical to $\mathbf{C}(x)$ on every index except for $j_1, j_2, \ldots$, where it takes on all value(s) with equal probability. Since locally decodable codes are tolerant of errors in fewer than δm positions, we have:

$$\begin{aligned} &\sum_{j \in [m]} \Pr_x[A(\mathbf{C}(x) \oplus \Delta_{j_1}, i) = x_i \mid A(\cdot, i) \text{ reads } j] \Pr[A(\cdot, i) \text{ reads } j] \\ &\qquad \geq 1/2 + \epsilon, \end{aligned}$$

where the probability is now taken over all coin tosses of A, all strings x, and all (random) vectors Δ_{j_1}. Again, this implies that there exists some index j_2 such that:

$$\Pr_x [A(\mathbf{C}(x) \oplus \Delta_{j_1}, i) = x_i \mid A(\cdot, i) \text{ reads } j_2] \geq 1/2 + \epsilon.$$

Note that we must have $j_2 \neq j_1$; this is so because the value at index j_1 of the codeword given as input to A is random and independent of the underlying data word x. Since the algorithm A does not know what is located in positions of the codeword other than the one it reads, we have:

$$\Pr_x [A(\mathbf{C}(x), i) = x_i \mid A(\cdot, i) \text{ reads } j_2] \geq 1/2 + \epsilon.$$

Or, in other words, j_2 is good for i.

Repeating this argument, we see that:

$$\begin{aligned} &\sum_{j \in [m]} \Pr_x[A(\mathbf{C}(x) \oplus \Delta_{j_1, j_2}, i) = x_i \mid A(\cdot, i) \text{ reads } j] \Pr[A(\cdot, i) \text{ reads } j] \\ &\qquad \geq 1/2 + \epsilon. \end{aligned}$$

Therefore, there exists an index j_3 such that:

$$\Pr_x\left[A(\mathbf{C}(x) \oplus \Delta_{j_1,j_2}, i) = x_i \mid A(\cdot, i) \text{ reads } j_3\right] \geq 1/2 + \epsilon,$$

so that j_3 is good for i with $j_3 \neq j_1, j_2$.

Since the code is tolerant of errors in up to δm positions, continuing in this manner shows that there is a set J_i of δm distinct indices, such that $j \in J_i$ implies j is good for i. But remember that this was for a fixed, arbitrary value i. Thus, for all $i \in [n]$ we have a set $J_i, |J_i| \geq \delta m$ such that $j \in J_i$ implies j is good for i.

By the pigeonhole principle, there must exist some $j^* \in [m]$ which is good for δn indices $i_1, \ldots, i_{\delta n} \in [n]$. Theorem 2 then shows that we must have $\log |\Sigma| \geq (1 - \mathcal{H}(1/2 + \epsilon))\delta n$. This gives the stated result. □

3.3 Lower Bounds for $q > 1$

In this section we prove lower bounds for locally decodable codes with $q > 1$. As mentioned previously, we find it convenient to work with, and to prove lower bounds for, the class of smooth codes. Recalling the earlier relationship between smooth codes and locally decodable codes as outlined in Theorem 1, we use this result to obtain our desired lower bound.

We seek to replicate the idea underlying the proof of Theorem 3. There, we sought a single index j^* which was *good* for a constant fraction of the indices of the data; that is, which gave information about a constant fraction of the input bits. Now we seek a subset S^* of the indices of the codeword such that knowledge of the values of all the indices in S^* gives information about a constant fraction of the input bits. Once we find such a subset, we apply Theorem 2 to obtain a lower bound on the length of the input.

Let $\mathbf{C} : \{0,1\}^n \rightarrow \Sigma^m$ be a (q, c, ϵ)-smooth code and let algorithm A be a (q, c, ϵ)-smooth decoding algorithm for $\mathbf{C}$. Let $s \subseteq [m]$. We say that a given invocation of A *reads* s if the set of indices which A reads in that invocation is exactly equal to the set s. Since A is restricted to read at most q indices in any invocation, we must have $|s| \leq q$ for any s which is read by A in some invocation. Analogous to the definition above, we say that s is ϵ-*good* for i (where $|s| \leq q$) if:

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

For all $i \in [n]$, define the hypergraph H_i as follows: H_i contains m vertices labeled by elements of $[m]$. The hyperedges of H_i, denoted E_i, are defined by:

$$E_i = \left\{ e \subseteq [m] \mid e \text{ is } \tfrac{\epsilon}{2}\text{-good for } i \right\}.$$

We say A *reads from* E_i (for brevity, A *reads* E_i) if A reads e and $e \in E_i$. Recall some standard terminology for hypergraphs: a *matching* in a hypergraph H with hyperedges E is defined as a set $M \subseteq E$ of hyperedges such that the intersection of any pair of (distinct) hyperedges in M is empty. A *vertex cover* for a hypergraph H is a set V of vertices such that every hyperedge in the set of hyperedges E contains at least one element of V. A standard result is that in a hypergraph whose hyperedges contain at most q vertices, if the size of the minimum vertex cover is at least $|V|$, then there is a matching of size at least $|V|/q$. With this in mind we now state the following lemma:

Lemma 4. *Let $\mathbf{C}$ be a (q, c, ϵ)-smooth code and $\{H_i\}_{i=1}^n$ be the associated set of hypergraphs as described above. For all i, H_i has a matching M_i of size at least $\epsilon m/cq$.*

Proof. Using the definition of a smooth code, we have:

$$\begin{aligned}
&1/2 + \epsilon \\
&\leq \Pr_x[A(\mathbf{C}(x), i) = x_i | A(\cdot, i) \text{ reads } E_i] \Pr[A(\cdot, i) \text{ reads } E_i] \\
&\quad + \Pr_x[A(\mathbf{C}(x), i) = x_i | A(\cdot, i) \text{ reads } E_i^c] \Pr[A(\cdot, i) \text{ reads } E_i^c] \\
&< \Pr[A(\cdot, i) \text{ reads } E_i] \\
&\quad + (1/2 + \epsilon/2)(1 - \Pr[A(\cdot, i) \text{ reads } E_i]),
\end{aligned}$$

where the second term of the inequality comes from the definition of the set of hyperedges E_i. This, in turn, implies that $\Pr[A(\cdot, i) \text{ reads from } E_i] > \epsilon$.

We associate with each hyperedge $e \in E_i$ the number P_e which is defined as the probability that $A(\cdot, i)$ reads e. Using this notation, we have $\sum_{e \in E_i} P_e > \epsilon$. Furthermore, since the code $\mathbf{C}$ is (q, c, ϵ)-smooth, for every $j \in [m]$ we have $\sum_{e \in E_i | j \in e} P_e \leq c/m$.

Let V be a vertex cover for H_i. Since for all $e \in E_i$ we have $e \cap V \neq \emptyset$ (by definition of a vertex cover), it follows from the previous statements that $\sum_{e \in E_i | e \cap V \neq \emptyset} P_e > \epsilon$. Then:

$$\epsilon < \sum_{e \in E_i | e \cap V \neq \emptyset} P_e \leq \sum_{j \in V} \sum_{e \in E_i | j \in e} P_e \leq |V| c/m,$$

which implies that the minimum vertex cover for H_i has size at least $\epsilon m/c$. This means that H_i has a matching of size at least $\epsilon m/cq$. □

Before stating our main result, we first state a technical lemma. Recall that M_i is a matching in H_i. The following lemma provides a bound on the number of vertices which must be chosen at random from H_i in order to have a constant probability of covering at least one hyperedge in M_i. More precisely, form a (multi)set S by choosing $|S|$ elements of the vertices of H_i at random, with replacement. Say that S *hits* M_i if there exists a subset $s \subseteq S$ such that $s \in M_i$.

Lemma 5. *Let H be a hypergraph on m vertices whose hyperedges all contain q or fewer vertices. Let H have a matching M of size γm ($\gamma < 1/q$). There exists $t = \Theta(\gamma^{-\frac{1}{q}} m^{\frac{q-1}{q}})$ such that for a collection S of size $|S| = t$ of randomly selected vertices of H:*

$$\Pr[S \text{ hits } M] > 3/4.$$

Proof. The worst case occurs when H is q-uniform; that is, each hyperedge in H (and hence in M) has exactly q vertices. This is because we can form a new matching M' by adding vertices to all those hyperedges in M containing fewer than q vertices (this can be done while maintaining a matching because we have $\gamma < 1/q$). The required probability is therefore lower bounded by the probability in the setting where H is q-uniform.

Let $S = \{v_1, \ldots, v_t\}$ be a collection of randomly chosen vertices of H. Let a be a q-element subset of $[t]$. Define the random variable Y_a as:

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

and furthermore define the random variable Y as the sum of the Y_a over all q-element subsets of $[t]$. According to this formulation, the quantity of interest can be expressed as:

$$\Pr[S \text{ hits } M] = \Pr[Y \neq 0].$$

We bound this probability by bounding the expectation and variance of Y. Since elements of S are chosen with replacement, all Y_a have the same distribution; thus:

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

Consider the auxiliary random variables $Z_a = Y_a - E[Y_a]$. The variance of Y can be calculated as:

$$\begin{aligned} \mathrm{Var}[Y] &= \mathrm{E}\left[\left(\sum_a Z_a\right)^2\right] \\ &= \sum_a \mathrm{E}\left[Z_a^2\right] + \sum_{a,a' \mid a \neq a'} \mathrm{E}\left[Z_a Z_{a'}\right]. \end{aligned} \tag{2}$$

The first term of (2) can be bounded as follows:

$$\begin{aligned} \mathrm{E}\left[Z_a^2\right] = \mathrm{Var}[Y_a] &= \mathrm{E}\left[Y_a^2\right] - \mathrm{E}^2[Y_a] \\ &< \mathrm{E}\left[Y_a^2\right] \\ &= \mathrm{E}[Y_a] = \frac{|M| q!}{m^q} = \frac{\gamma m q!}{m^q}. \end{aligned}$$

As for the second term of (2), note that if $a \cap a' = \emptyset$, then Z_a and $Z_{a'}$ are independent, so that $\mathrm{E}[Z_a Z_{a'}] = \mathrm{E}[Z_a]\mathrm{E}[Z_{a'}] = 0$. On the other hand, if the intersection of a and a' is nonempty, since M is a matching we cannot have both $\{v_{a_1}, \ldots, v_{a_q}\}, \{v_{a'_1}, \ldots, v_{a'_q}\} \in M$, and therefore $E[Y_a Y_{a'}] = 0$. Thus:

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

Putting everything together, we see that:

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

Using Chebyshev's inequality gives:

$$\begin{aligned} \Pr[Y = 0] &\leq \Pr[|Y - \mathrm{E}[Y]| \geq \mathrm{E}[Y]] \\ &\leq \mathrm{Var}[Y]/\mathrm{E}^2[Y] \\ &< \frac{m^q}{\gamma m \binom{t}{q} q!}, \end{aligned}$$

which is bounded above by $1/4$ for $t = \Theta(\gamma^{-\frac{1}{q}} m^{\frac{q-1}{q}})$. $\square$

Using the previously developed lemmas, we are now able to state and prove our main result:

THEOREM 6. *Let* $\mathbf{C} : \{0,1\}^n \to \Sigma^m$ *be a* (q, δ, ϵ)*-locally decodable code. Then:*

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

PROOF. Theorem 1 shows that $\mathbf{C}$ is $(q, q/\delta, \epsilon)$-smooth. Lemma 4 and the discussion which precedes it show that, for each $i \in [n]$, there exists a set M_i of *disjoint* q-or-fewer-tuples of $[m]$ such that:

- For all $e \in M_i$,

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

- The size of M_i is at least $\epsilon\delta m/q^2$.

Recall that we consider each set M_i as a matching in a hypergraph with m vertices. Lemma 5 shows the existence of some set S^* of size $t = \Theta((\epsilon\delta/q^2)^{-\frac{1}{q}} m^{\frac{q-1}{q}})$ such that for $3n/4$ indices $i_1, \ldots, i_{3n/4} \in [n]$, S^* hits each of $M_{i_1}, \ldots, M_{i_{3n/4}}$. Restated, this means that reading from codeword $\mathbf{C}(x)$ these t elements (which are elements of Σ) gives information about $3n/4$ bits of the original data. Theorem 2 then shows that $t \log|\Sigma| \geq \frac{3n(1-\mathcal{H}(1/2+\epsilon))}{4}$, and this gives the result stated in the Theorem. $\square$

4. EXTENSIONS AND OPEN QUESTIONS

We briefly point out some extensions and further applications of our results.

Adaptive versus non-adaptive queries.

One open question concerns lower bounds for locally decodable/smooth codes when the decoding procedure is allowed to make adaptive queries (for the case $q \geq 2$). We see no fundamental reason why the lower bounds given above should not hold, but the given proof fails in the consideration of the adaptive case. We note, however, that any adaptive (q, δ, ϵ)-local decoding algorithm A can be converted into a non-adaptive $(\frac{|\Sigma|^q - 1}{|\Sigma| - 1}, \delta, \epsilon)$-decoding algorithm A' for the same code: simply have A' toss the same coins as A, and then have A' make all possible queries depending on the $|\Sigma|^{q-1}$ possible values of the first $q-1$ queries. Similarly, an adaptive (q, δ, ϵ)-local decoding algorithm A can be converted into a non-adaptive $(q, \delta, \epsilon/|\Sigma|^{q-1})$-local decoding algorithm A' by having A' "guess" the values of the codeword in the first $q-1$ indices queried and submitting a set of queries based upon this guess. If A' guesses correctly, the decoding procedure works with probability $1/2 + \epsilon$; otherwise, A' outputs a random bit which is correct with probability $1/2$. Better reductions, which would imply better lower bounds, should be possible.

Smooth encodings versus locally-decodable encodings.

As the counterpart to Theorem 1, we note that the converse to the theorem also holds. That is, any (q, c, ϵ)-smooth decoding algorithm can be used as a $(q, \delta, \epsilon - c\delta)$-local decoding algorithm (since the probability of reading from only non-corrupted entries is at least $(1 - c\delta)$). Again, perhaps a better reduction is possible.

Smooth encodings and PIR.

It is interesting to note that the arguments presented in Section 3.3 above indicate that any (q, c, ϵ)-smooth decoding algorithm A (with $c > q$) can be converted into a $(q, q, \epsilon^2/2c)$-smooth decoding algorithm A' where the queries made by A' have uniform distribution. The construction uses the matchings M_i shown to exist by Lemma 4; additional hyperedges are formed with the remaining vertices (while maintaining a matching) in an arbitrary fashion to form M'_i. $A'(\cdot, i)$ simply picks a hyperedge at random from M'_i; with probability ϵ/c the hyperedge is good for i and the success probability is greater than $1/2 + \epsilon/2$. Otherwise, A' outputs a random bit and has success probability $1/2$. When a (q, q, ϵ)-smooth encoding $\mathbf{C} : \{0,1\}^n \to \Sigma^m$ has a decoding procedure where each query is asked according to the uniform distribution, the encoding gives an information-theoretic PIR scheme (see [7]) with q servers, query size $\log m$, and answer size $\log |\Sigma|$, such that the user has probability $1/2 + \epsilon$ of correctly retrieving the bit he is interested in. There is also a reverse connection. A PIR scheme with q servers, query size t, answer size l, and probability of correct retrieval $1/2 + \epsilon$ can be turned into a system where each query is uniformly distributed, the query size is $t + \log O(1/\epsilon)$, and the probability of correct retrieval is $1/2 + \epsilon/2$. This in turn gives a $(q, q, \epsilon/2)$-smooth encoding $\mathbf{C} : \{0,1\}^n \to (\{0,1\}^l)^m$ where $m = O(q2^t/\epsilon)$. These connections show that the problem of finding lower bounds for information-theoretic PIR is essentially equivalent to the problem of finding lower bounds for locally decodable codes. An important difference is that in locally decodable codes one is interested in the case of encodings over small alphabets, while in PIR the efficiency measure is given by $\log m + \log |\Sigma|$ (and so one is interested in the case where $m \approx |\Sigma|$).

Acknowledgments

We thank Oded Goldreich for several insightful discussions. L. T. would like to thank Leonid Levin for suggesting the problem.

5. REFERENCES

[1] A. Ambainis. Upper bounds on the communication complexity of private information retrieval. In *Proceedings of ICALP*, pages 401–407, 1997.

[2] S. Arora, C. Lund, R. Motwani, M. Sudan, and M. Szegedy. Proof verification and hardness of approximation problems. *Journal of the ACM*, 45(3):501–555, 1998. Preliminary version in *Proc. of FOCS'92*.

[3] S. Arora and S. Safra. Probabilistic checking of proofs: A new characterization of NP. *Journal of the ACM*, 45(1):70–122, 1998. Preliminary version in *Proc. of FOCS'92*.

[4] L. Babai, L. Fortnow, L. Levin, and M. Szegedy. Checking computations in polylogarithmic time. In *Proceedings of the 23rd ACM Symposium on Theory of Computing*, pages 21–31, 1991.

[5] L. Babai, L. Fortnow, N. Nisan, and A. Wigderson. BPP has subexponential time simulations unless EXPTIME has publishable proofs. *Computational Complexity*, 3(4):307–318, 1993.

[6] D. Beaver and J. Feigenbaum. Hiding instances in multi-oracle queries. In *7th Annual Symposium on Theoretical Aspects of Computer Science*, volume 415 of *Lecture Notes in Computer Science*, pages 37–48, Rouen, France, 22–24 February 1990. Springer.

[7] B. Chor, O. Goldreich, E. Kushilevitz, and M. Sudan. Private information retrieval. *Journal of the ACM*, 45(6), 1998. Preliminary version in *Proc. of FOCS'95*.

[8] J. Feigenbaum and L. Fortnow. Random self-reducibility of complete sets. *SIAM Journal on Computing*, 22(5):994–1005, October 1993.

[9] P. Gemmell, R. Lipton, R. Rubinfeld, M. Sudan, and A. Wigderson. Self-testing/correcting for polynomials and for approximate functions. In *Proceedings of the Twenty Third Annual ACM Symposium on Theory of Computing*, pages 32–42, New Orleans, Louisiana, 6–8 May 1991.

[10] P. Gemmell and M. Sudan. Highly resilient correctors for polynomials. *Information Processing Letters*, 43(4):169–174, 28 September 1992.

[11] O. Goldreich. Personal communication. September 1999.

[12] L. Levin. Research overview. `http://www.cs.bu.edu/fac/lnd/research/res.html`.

[13] R. Lipton. New directions in testing. In *Proceedings of DIMACS Workshop on Distributed Computing and Cryptography*, 1989.

[14] E. Mann. Private access to distributed information. Master's Thesis, Technion, 1998.

[15] A. Polishchuk and D.A. Spielman. Nearly linear-size holographic proofs. In *Proceedings of the 26th ACM Symposium on Theory of Computing*, pages 194–203, 1994.

[16] M. Sudan, L. Trevisan, and S. Vadhan. Pseudorandom generators without the XOR lemma. In *Proceedings of the 31st ACM Symposium on Theory of Computing*, pages 537–546, 1999.