THEORY OF COMPUTING, Volume 10 (19), 2014, pp. 515–533
www.theoryofcomputing.org

Query Complexity Lower Bounds for Reconstruction of Codes

Sourav Chakraborty Eldar Fischer* Arie Matsliah

Received June 10, 2012; Revised August 25, 2014; Published December 29, 2014

Abstract: We investigate the problem of *local reconstruction*, as defined by Saks and Seshadhri (2008), in the context of error correcting codes.

The first problem we address is that of *message reconstruction*, where given oracle access to a corrupted encoding $w \in \{0,1\}^n$ of some message $x \in \{0,1\}^k$ our goal is to probabilistically recover x (or some portion of it). This should be done by a procedure (reconstructor) that given an index i as input, probes w at few locations and outputs the value of x_i. The reconstructor can (and indeed must) be randomized, with all its randomness specified in advance by a single random seed, and the main requirement is that for *most* random seeds, *all* values $x_1, \ldots, x_k$ are reconstructed correctly. (Notice that swapping the order of "for most random seeds" and "for all $x_1, \ldots, x_k$" would make the definition equivalent to standard *Local Decoding*.)

A message reconstructor can serve as a "filter" that allows evaluating certain classes of algorithms on x safely and efficiently. For instance, to run a parallel algorithm, one can initialize several copies of the reconstructor with the same random seed, and then they can autonomously handle decoding requests while producing outputs that are consistent with the

A conference version of this paper appeared in the The Proceedings of the Second Symposium on Innovations in Computer Science (ICS 2011) [6].

*Supported by an ERC-2007-StG grant number 202405.

ACM Classification: F.2.2, H.1.1

AMS Classification: 68Q17, 68W20

Key words and phrases: property testing, locally decodable codes, locally correctable codes, reconstruction

 DOI: 10.4086/toc.2014.v010a019

original message x. Another motivation for studying message reconstruction arises from the theory of Locally Decodable Codes.

The second problem that we address is *codeword reconstruction*, which is similarly defined, but instead of reconstructing the message the goal is to reconstruct the codeword itself, given oracle access to its corrupted version.

Error correcting codes that admit message and codeword reconstruction can be obtained from Locally Decodable Codes (LDC) and Self Correctable Codes (SCC) respectively. The main contribution of this paper is a proof that in terms of query complexity, these constructions are close to best possible in many settings, even when we disregard the length of the encoding.

1 Introduction

Consider the following problem: a large data set $x \in \{0,1\}^k$ is stored on a storage device in encoded form, but a small fraction of the encoding may be corrupted. We want to execute an algorithm $\mathcal{M}$ on x, but most likely $\mathcal{M}$ will only need a small fraction of x for its execution. This can be the case if $\mathcal{M}$ is a single process in a large parallelized system, or if $\mathcal{M}$ is a querying algorithm with limited memory, that can even be adaptive, not knowing which bits of the input it will need in advance. Ideally, in all these cases $\mathcal{M}$ should have the ability to *efficiently* decode any bit of x only when the need arises (in particular, decoding every bit should be done by reading only a small fraction of the corrupted encoding), and to ensure correctness it is also necessary that $\mathcal{M}$ succeeds (with high probability) in correctly decoding *all* bits that are required for its execution. One way of ensuring this is to decode the whole message x in advance, but in many cases this may be very inefficient, or even impossible.

To perform the above task, a *message reconstructor* is required, that can simulate query access to x in a local manner. Concretely, a *message reconstructor* is an algorithm that can recover the original message $x \in \{0,1\}^k$ from a corrupted encoding $w \in \{0,1\}^n$ under two main conditions: *(locality)* for every $i \in [k]$ reconstructing x_i requires reading w only at very few locations; *(consistency)* with high probability, *all* indices $i \in [k]$ should be reconstructed correctly. When the consistency condition is weakened, so that only each index in itself is reconstructed correctly with high probability, the definition becomes equivalent to *Local Decoding*, as formally defined in [15]. Informally, a locally decodable code (LDC) is an error-correcting code which allows to probabilistically decode any symbol of an encoded message by probing only a few symbols of its corrupted encoding. As in the case of LDCs, the main challenge in message reconstruction is to find short codes that allow reconstruction of every x_i with as few queries to w as possible.

Any q-query LDC can be used for $O(q\log k)$-query message reconstruction, by simply repeating the local decoding procedure $O(\log k)$ times (for every decoding request x_i) and outputting the majority vote. The repetition will reduce the probability of incorrectly reconstructing x_i to $1/(3k)$, so that with probability $2/3$ all k indices are reconstructed correctly. Thus having $O(1)$-query LDCs (e. g., the Hadamard code) immediately implies the existence of codes with an $O(\log k)$-query message reconstructor. The question that immediately arises is whether one can do better, specifically in terms of query complexity. The first theorem of this paper (see Theorem 3.2) states that for any encoding of any length, a non-adaptive message reconstructor must make $\Omega(\log k)$ queries per decoding request.

Another family of error correcting codes related to LDCs are *self-correctable codes* (SCC) [5]. In a q-query self-correctable code the probabilistic decoder is required to recover an arbitrary symbol of the encoding itself. The second problem that we study here is of *codeword reconstruction*, which is related to SCCs in the same manner that message reconstruction is related to LDCs. Concretely, a *codeword reconstructor* is an algorithm that can recover a codeword $y \in \{0,1\}^n$ from its corrupted version $w \in \{0,1\}^n$ with two conditions: *(locality)* for every $i \in [n]$ reconstructing y_i requires reading w only at very few locations; *(consistency)* with high probability, *all* indices $i \in [n]$ should be reconstructed correctly. Here too, if the consistency condition is weakened, so that only each index in itself is reconstructed correctly with high probability, then the definition becomes equivalent to self-correction and any q-query SCC can be used for $O(q \log n)$-query codeword reconstruction. Thus having $O(1)$-query SCCs (e. g., the Hadamard code) immediately implies the existence of an $O(\log n)$-query codeword reconstructor.

The second theorem of this paper (see Theorem 3.4) gives lower bounds on the query complexity of codeword reconstruction for *linear* codes.[1] Denoting by $\hat{n}$ the number of distinct rows in the generating matrix of a linear code, Theorem 3.4 states that codeword reconstruction requires $\Omega(\sqrt{\log \hat{n}})$ queries per decoding request. Since essentially all known SCCs are linear codes with $\hat{n} = n$, this bound is tight up to the square root. Furthermore, a lower bound on the query complexity can neither be stated for general (non-linear) codes, nor stated in terms of n alone. We elaborate on this in Remark 3.5 below.

1.1 Related work

1.1.1 Local reconstruction

Initially the model of *online property reconstruction* was introduced in [1]. In this setting a data set f is given (we can think of it as a function $f : [n]^d \to \mathbb{N}$) which should have a specified structural property P, but this property may not hold due to unavoidable errors. The specific property studied in [1] was monotonicity, and the goal of the reconstructor was to construct (online) a new function g that is monotone, but not very far from the original f. The authors developed such a reconstructor, which given $x \in [n]^d$ could compute $g(x)$ by querying f at only few locations. However, the reconstructor had to "remember" its previous answers in order to be consistent with some fixed monotone function g, making it not suitable for parallel or memory-limited applications.

This issue was addressed in [17], where the authors presented a purely local reconstructor for monotonicity, that could output $g(x)$ based only on few inspections of f. Given a random string r, the reconstructor of [17] could locally reconstruct g at any point x, such that all answers were consistent with some function g, which for most random strings r was monotone. Such a reconstructor affords an obvious distributed implementation: generate one random seed, and distribute it to each of the copies of the reconstructor. Since they are all determined by r, their answers will be consistent.

Local reconstruction was also studied in the context of graphs [14, 12], functions on discrete domains [4, 13] and geometric problems [7]. In particular, [14] studied the problem of expander reconstruction. Given an oracle access to a graph that is close to being an expander, the algorithm of [14] can simulate an oracle access to a corrected graph, that is an expander. Reconstruction in the context of partition problems for dense graphs was implicitly studied in [12]. Given a dense graph G that is close to

[1] An code $\mathcal{C}$, encoding a k bit string by an n bit string, is *linear* if (for all k) it has a *generating matrix* $G \in \{0,1\}^{n \times k}$ such that for all $x \in \{0,1\}^k$, $\mathcal{C}(x) = Gx$.

satisfying some partition property (say k-colorability), the approximate partitioning algorithm from [12] can be made into one that efficiently and locally reconstructs a graph G' that satisfies the partition property and is close to G.

1.1.2 LDCs and SCCs

Locally decodable codes were explicitly defined in [15], but they were extensively studied before that in the context of self-correcting computation, worst-case to average-case reductions, randomness extraction, probabilistically checkable proofs and private information retrieval (see [18] for a survey). The initial constructions of LDCs (and SCCs) were based on Reed-Muller codes and allowed to encode a k-bit message by a $\mathrm{poly}(\log k)$-query LDC of length $\mathrm{poly}(k)$ [3, 8].

For a fixed number of queries, the complexity of LDCs was first studied in [15], and has since been the subject of a large body of work (see [18, 11] for surveys). To this day, there is a nearly exponential gap between the best known upper bounds on the length of q-query LDCs [21, 9] and the corresponding lower bounds [16, 19, 20], for any constant $q \geq 3$.

While interesting in its own right, studying message reconstruction can help us in understanding the limitations of LDCs. If we view the local decoding algorithm as a deterministic algorithm that takes as input $i \in [k]$ and a random string r, then we require that for each i, most choices of r lead to the correct decoding of the ith bit of the message. For message reconstruction we swap the *for all i* and *for most r* quantifiers, and require that for most choices of r, the algorithm correctly decodes all k bits of the encoded message.

Our lower bounds imply that in the case of error-correcting codes, it is impossible to correlate the success probabilities of a local decoder in a way that for most random strings r, either all bits of the message are decoded correctly, or only very few of them are. This is in contrast to the results from local reconstruction of general properties (described in previous section), where clever use of the fixed randomness r allows reconstruction with very few queries.

As we explained earlier, a constant query LDC of polynomial length would imply the existence of an $O(\log k)$-query message reconstructor of polynomial rate. Thus constructing an $O(\log k)$-query message reconstructable code of polynomial rate can be an intermediate step towards constant-query LDCs of polynomial length.

2 Preliminaries

For $\alpha \in \Sigma^n$ we denote by α_i the i-th symbol of α, and for a subset $I \subseteq [n]$ we denote by $\alpha\restriction_I$ the restriction of α to the indices in I. The *Hamming distance*, or simply the *distance* $d(\alpha,\beta)$ between two strings $\alpha,\beta \in \Sigma^n$, is the number of indices $i \in [n]$ such that $\alpha_i \neq \beta_i$. Given a set $S \subseteq \Sigma^n$ and $\alpha \in \Sigma^n$, the distance of α from S is defined as $d(\alpha,S) = \min_{\beta \in S}\{d(\alpha,\beta)\}$, where as expected the minimum over an empty set is defined to be $+\infty$. For $\varepsilon > 0$ we say that α is *ε-close* to S if $d(\alpha,S) \leq \varepsilon n$.

An $[n,k,d]_\Sigma$ code is a mapping $C : \Sigma^k \to \Sigma^n$ such that for every $x \neq x' \in \Sigma^k$, $d(C(x),C(x')) \geq d$. The parameter k is called the *message length* (or the *information length*) of the code and n is called the *block length* or simply *length* of the code. Sometimes we refer to C also as a subset of Σ^n defined as $C = \{y : \exists x \in \Sigma^k, y = C(x)\}$, and the elements $y \in C$ are called *codewords*.

Usually we will be interested in some family $\mathcal{C}$ of codes $\langle C_k\rangle_{k\in\mathbb{N}}$, with functions $n = n(k)$ and $d = d(k)$, in which every C_k is an $[n(k),k,d(k)]_\Sigma$ code. Whenever the alphabet Σ is not specified it means that $\Sigma = \{0,1\}$. With a slight abuse of notation, for $x \in \Sigma^k$ we will denote by $\mathcal{C}(x)$ the mapping $C_k(x)$ given by the "right size" code C_k from the family $\mathcal{C}$. Similarly, for $n = n(k)$ and $w \in \Sigma^n$ we define the distance of w from $\mathcal{C}$ as $d(w,\mathcal{C}) = d(w,C_k)$, which is the minimal distance between w and some codeword of $\mathcal{C}$. If $d(w,\mathcal{C}) < d/2$ then the minimum is achieved by a unique codeword $y \in \mathcal{C}$, and we denote this codeword by $D_{\mathcal{C}}(w)$. In this case the original message is well defined, and it will be denoted by $\mathcal{C}^{-1}(D_{\mathcal{C}}(w))$.

An $[n,k,d]$ code $\mathcal{C}$ is *linear* if (for all k) it has a *generating matrix* $G \in \{0,1\}^{n\times k}$ such that[2] for all $x \in \{0,1\}^k$, $\mathcal{C}(x) = Gx$. We denote by $\mathrm{R}(\mathcal{C})$ the number of distinct non-zero rows in $\mathcal{C}$'s generating matrix.

3 Definition of our model and statement of main results

Here we formally define message and codeword reconstruction, and state our main results. All our definitions apply to non-adaptive algorithms, i. e., algorithms that base their query strategy solely on their random bits, while basing their output on both random bits and the answers to the queries. For clarity of analysis we make the dependence on a random string r of bits (assumed to be chosen uniformly and independently) explicit, and we restrict[3] our attention to families of $[n,k,d]$ codes, where $\Sigma = \{0,1\}$. We disregard computation time considerations because all lower bounds presented here hold regardless of computation time.

Definition 3.1 (Message Reconstruction)**.** Let $\mathcal{C}$ be a family of $[n,k,d]$ codes with $d \geq 2\delta n$ for some fixed $\delta > 0$, let $q,\rho : \mathbb{N} \to \mathbb{N}$ and let ε be a fixed constant[4] satisfying $0 < \varepsilon < \delta$. A (q,ε,ρ) *message reconstructor for* $\mathcal{C}$ is a deterministic machine $\mathcal{A}$, taking as inputs $k, n = n(k) \in \mathbb{N}$, $i \in [k]$ and a random string $r \in \{0,1\}^{\rho(k)}$, that for every $w \in \{0,1\}^n$ satisfies the following:

- $\mathcal{A}$ generates a set $Q_{i,r} \subseteq [n]$ of $q = q(k)$ indices and a function $C_{i,r} : \{0,1\}^q \to \{0,1\}$, and outputs
$$\mathcal{A}^w_r(i) \triangleq C_{i,r}\left(w\restriction_{Q_{i,r}}\right).$$
- Let $\mathcal{A}^w_r \in \{0,1\}^k$ denote the concatenation $\mathcal{A}^w_r(1)\mathcal{A}^w_r(2)\cdots\mathcal{A}^w_r(k)$. If $d(w,\mathcal{C}) \leq \varepsilon n$ then
$$\Pr_r\left[\mathcal{A}^w_r = \mathcal{C}^{-1}(D_{\mathcal{C}}(w))\right] \geq 2/3\,.$$

When it is not important, we will avoid mentioning explicitly the random string length ρ, and will simply use the term (q,ε) *message reconstructor* (or even *q-query message reconstructor*) to mean a $(q,\varepsilon,\Omega(1))$ *message reconstructor*. For abbreviation, we may also use $\mathcal{A}_r$ to denote the algorithm $\mathcal{A}$ that operates with the fixed random string $r \in \{0,1\}^\rho$.

[2]Here and in the following we may identify $\{0,1\}$ with the field of two elements in the standard way.

[3]Nevertheless, the bounds presented in this paper generalize to any constant size alphabets as well.

[4]For clarity of presentation ε will be considered to be an absolute constant. Nevertheless, the bounds presented here have only logarithmic dependence on $1/\varepsilon$.

In this terminology, if a code $\mathcal{C}$ has a (q,ε) message reconstructor then it is locally decodable with q queries, up to noise rate ε. On the other hand, a code that is locally decodable with q queries (up to noise rate ε) has an $(O(q\log k),\varepsilon)$ message reconstructor, and so the existence of constant query LDCs implies that there exist codes that have an $(O(\log k),\Omega(1))$ message reconstructor. Our first result shows that in terms of the number of queries this is essentially optimal, even for arbitrarily long codes.

Theorem 3.2. *There are no codes (of* any *length) with an $o(\log k)$-query non-adaptive message reconstructor.*

Next we formally define codeword reconstruction. Notice that here the query complexity and the randomness of the algorithm are mentioned in terms of n, the block length, rather than k as in the definition of message reconstruction.

Definition 3.3 (Codeword Reconstruction)**.** Let $\mathcal{C}$, δ, q, ρ and ε be as in Definition 3.1. A (q,ε,ρ) *codeword reconstructor for* $\mathcal{C}$ is a deterministic machine $\mathcal{A}$, taking as inputs $k,n=n(k)\in\mathbb{N}$, $i\in[n]$ and a random string $r\in\{0,1\}^{\rho(n)}$, that for every $w\in\{0,1\}^n$ satisfies the following conditions:

- $\mathcal{A}$ generates a set $Q_{i,r}\subseteq[n]$ of $q=q(n)$ indices and a function $C_{i,r}:\{0,1\}^q\to\{0,1\}$, and outputs
$$\mathcal{A}_r^w(i)\triangleq C_{i,r}\left(w\restriction_{Q_{i,r}}\right).$$
- Let $\mathcal{A}_r^w\in\{0,1\}^n$ denote the concatenation $\mathcal{A}_r^w(1)\mathcal{A}_r^w(2)\cdots\mathcal{A}_r^w(n)$. If $d(w,\mathcal{C})\le\varepsilon$ then
$$\Pr_r\left[\mathcal{A}_r^w=D_{\mathcal{C}}(w)\right]\ge 2/3\,.$$

Here too, we may avoid mentioning ρ explicitly, and may use $\mathcal{A}_r$ to denote the algorithm $\mathcal{A}$ that operates with the fixed random string r.

If a code $\mathcal{C}$ has a (q,ε) codeword reconstructor then it is self-correctable with q queries, and conversely, any code that is self-correctable with q queries up to noise rate ε has an $(O(q\log n),\varepsilon)$ codeword reconstructor. As stated earlier, constant query SCCs exist, hence there are codes with an $(O(\log n),\Omega(1))$ codeword reconstructor. Since essentially all known LDCs and SCCs are linear codes, and furthermore they satisfy $\mathrm{R}(\mathcal{C})=n$ (i. e., all rows in their generating matrix are distinct), we actually have linear codes with an $(O(\log\mathrm{R}(\mathcal{C})),\Omega(1))$ codeword reconstructor. Our second result shows that in the case of linear codes one cannot get significantly better than that.

Theorem 3.4. *There are no linear codes (of* any *length) with an $o(\sqrt{\log\mathrm{R}(\mathcal{C})})$-query non-adaptive codeword reconstructor.*

Remark 3.5. For general (non-linear) codes there is no lower bound on the query complexity which is poly-logarithmic in n. This follows from a simple observation that if a code has a q-query message reconstructor then it also has a qk-query codeword reconstructor (in qk queries it is possible to decode the whole message and its encoding). So, for example, Long Codes admit a codeword reconstructor that makes only $O(k\log k)=O(\log\log n\log\log\log n)$ queries.

Furthermore, even if we focus on linear codes only, stating the bound in Theorem 3.4 in terms of n (instead of $\mathrm{R}(\mathcal{C})$) is impossible, since there are linear $[n,k,d]$ codes that have a q-query codeword reconstructor, with q arbitrarily smaller than n. For example, let $\mathcal{C}'$ be a linear $[n',k,d']$ code with a (q,ε) codeword reconstructor, and define a new linear $[n=tn',k,d=td']$ code $\mathcal{C}$ as $\mathcal{C}(x)=\mathcal{C}'(x)\cdots\mathcal{C}'(x)$, namely, encoding x with t copies of $\mathcal{C}'(x)$. Now for any t (and hence arbitrarily large n), $\mathcal{C}(x)$ has a $(q,\varepsilon/2)$ codeword reconstructor, which picks a random copy of $\mathcal{C}'(x)$ from the encoding, and then simulates on it the original codeword reconstructor for $\mathcal{C}'$.

In the following corollary we use the fact that any linear code can be transformed into a systematic code[5] and combine Theorem 3.2 with Theorem 3.4.

Corollary 3.6. *There is no linear code with an $o(\max\{\log k, \sqrt{\log \mathrm{R}(\mathcal{C})}\})$-query codeword reconstructor.*

It is worth mentioning that, while both local decoding and self-correction become trivial in the random-noise model (via simple repetition codes), this is not the case for reconstruction. In fact, the query-complexity lower bounds from Theorems 3.2 and 3.4 apply to the random-noise model as well. See more details in Section 6.1.

4 Proof of Theorem 3.2

Outline: Usually, lower bound proofs that use Yao's Principle involve an input distribution that fools any deterministic algorithm of a certain type (bounded query complexity in our case) with probability larger than $1/3$. However, in this proof we will need to analyse the probabilistic algorithm $\mathcal{A}$ itself, giving special treatment to the indices that it queries with high probability. We first prove that for most of the deterministic algorithms $\mathcal{A}_r$ that result from fixing the random seed r in $\mathcal{A}$ (the "typical" ones), the number of bits that are reconstructed correctly based only on the indices that are frequently queried by $\mathcal{A}$ is small. Then we show that there is a distribution that fools any such typical algorithm with very high probability. Due to this technique, instead of the usual application of Yao's Principle we argue that there is a distribution $\mathcal{D}$ that fools at least half of the deterministic algorithms $\mathcal{A}_r$ with probability $1-o(1)$. Thus, the distribution $\mathcal{D}$ would fool the probabilistic algorithm $\mathcal{A}$ with probability at least $2/5-o(1)$.

Fix $\varepsilon>0$. Let $\mathcal{C}$ be a family of $[n,k,d]$ codes and let $\mathcal{A}$ be its (q,ε) message reconstructor. Our goal is to prove that $q=\Omega(\log k)$. Recall that $Q_{i,r}\subseteq[n]$ denotes the set of indices queried by $\mathcal{A}$ on input $i\in[k]$, given the random string r. For $j\in[n]$ we define

$$I(j,r)=\{i\in[k]: j\in Q_{i,r}\}\ .$$

Namely, $I(j,r)$ is the set of indices whose reconstructed value may depend on the j-th bit of the received word, given that the random seed is r. Similarly, for a subset $S\subset[n]$ we define

$$I(S,r)=\bigcup_{j\in S} I(j,r)=\{i\in[k]: S\cap Q_{i,r}\neq\emptyset\}\ .$$

We call an index $j\in[n]$ *influential* with respect to r if $|I(j,r)|>10q$, and *non-influential* otherwise.

[5] A linear code $\mathcal{C}$ is systematic if $\mathcal{C}(x)\restriction_{[k]}=x$ for all $x\in\{0,1\}^k$.

The following two lemmas follow by considering the bipartite graph G_r with message indices on the left and code indices on the right, where edges are defined by $Q_{i,r}$ and $I(j,r)$, i. e., ij is an edge if and only if $j \in Q_{i,r}$ if and only if $i \in I(j,r)$.

Lemma 4.1. *For any r, there are at most $k/10$ influential indices in $[n]$.*

Proof. If there are more than $k/10$ influential indices then

$$\sum_{j=1}^{n} |I(j,r)| > \left(\frac{k}{10}\right)(10q) = kq\,.$$

On the other hand

$$\sum_{j=1}^{n} |I(j,r)| = \sum_{i=1}^{k} |Q_{i,r}| \leq kq\,,$$

a contradiction. □

Lemma 4.2. *Let $A_r^0 \subseteq [n]$ be the set of influential indices (with respect to r). There exists a partition $A_r^1, A_r^2, \ldots, A_r^T$ of $[n] \setminus A_r^0$ into $T \leq k$ parts such that for all $i \in [T]$, $|I(A_r^i, r)| \leq 10q$.*

Proof. Recall that $\sum_{j=1}^{n} |I(j,r)| \leq kq$ from the previous proof. Let us construct a partition $A_r^1, \ldots, A_r^T$ of the non-influential indices as follows: A_r^1 contains the first ℓ_1 non-influential indices, where ℓ_1 is the maximal number for which $|I(A_r^1)| \leq 10q$ (note that in particular $\ell_1 \geq 1$); then A_r^2 contains the next ℓ_2 non-influential indices, where ℓ_2 is the largest number satisfying $|I(A_r^2)| \leq 10q$ and so on. We claim that in the end of this process $T \leq k$.

Assume that $T > k$. Consider the partition $B^1, \ldots, B^{\lceil T/2 \rceil}$ of the non-influential indices where $B^i = A_r^{2i-1} \cup A_r^{2i}$. Notice that since $T \geq k+1$, there are at least $k/2$ parts in this partition. By the definition of the A_r^i, all but at most one (the last one) of the B^i satisfy $I(B^i) > 10q$. So we have

$$\sum_{j=1}^{n} |I(j,r)| \geq \sum_{i=1}^{\lceil T/2 \rceil - 1} |I(B^i, r)| > (k/2-1)10q > kq$$

contradicting the fact $\sum_{j=1}^{n} |I(j,r)| \leq kq$. □

Now let us define a distribution $\mathcal{D}$ over received words. We will use the notation $x \sim \mathcal{D}$ to mean that x is chosen at random according to $\mathcal{D}$, and whenever D is a set, we also use $x \sim D$ to mean that x is chosen uniformly at random from D. Whenever the distribution or the set are clear from context we may omit their specification.

Recall that we need a distribution over received words that are ε-close to $\mathcal{C}$, on which every deterministic message reconstructor fails with probability larger than $1/3$. Instead, we define a distribution $\mathcal{D}$ that will provide a word that is ε-close to $\mathcal{C}$ only with probability $1-o(1)$. However, this will be sufficient because we will show that any algorithm will with probability at least $2/5-o(1)$ fail to reconstruct the correct message. So also if we condition on the probability $1-o(1)$ event we still get the same error probability estimate.

A random word $w \sim \mathcal{D}$ is generated by picking a uniformly random $x \sim \{0,1\}^k$, setting $y = \mathcal{C}(x)$ and then obtaining w by flipping each of the bits of y with probability $\varepsilon/2$, independently of the other bits. In fact this will be a distribution over w and x, but with some abuse of notation we will generally omit x, as with probability $1-o(1)$ it is just the message corresponding to the codeword closest to w. We need the following simple fact about $\mathcal{D}$.

Fact 4.3. *For every $q \le n$, $Q \subseteq [n]$ of size $|Q| = q$ and $\alpha \in \{0,1\}^q$,*

$$\Pr_{w\sim\mathcal{D}}\left[w\restriction_Q = \alpha\right] \ge (\varepsilon/2)^q\,.$$

We shall prove that if the query complexity of $\mathcal{A}$ is $o(\log k)$, then the probability that $\mathcal{A}$ fails on $w \sim \mathcal{D}$ is large, namely,

$$\Pr_{w\sim D,r}\left[\mathcal{A}_r^w = \mathcal{C}^{-1}(D_{\mathcal{C}}(w))\right] \le 1/2 + o(1)\,.$$

For $w \sim \{0,1\}^n$, $r \in \{0,1\}^\rho$ and $i \in [k]$ we say that $\mathcal{A}_r^w(i)$ is *determined by* $w\restriction_{A_r^0}$ if $\mathcal{A}_r^w(i) = \mathcal{A}_r^y(i)$ for every $y \in \{0,1\}^n$ with $y\restriction_{A_r^0} = w\restriction_{A_r^0}$ (notice that in general, $\mathcal{A}_r^w(i)$ may be determined by $w\restriction_{A_r^0}$ even when $Q_{i,r} \nsubseteq A_r^0$). The next definition and lemma say that for most random strings r, the expected number of indices correctly determined only by the influential indices is not very large, where the expectation is taken over $w \sim \mathcal{D}$.

Definition 4.4. For every $x \in \{0,1\}^k$, $w \in \{0,1\}^n$ and $r \in \{0,1\}^\rho$ we set $\beta_r^{w,x} = 0$ if there is any index i such that $\mathcal{A}_r^w(i)$ is determined by $w\restriction_{A_r^0}$ but does not match the value x_i. If there is no such index, then we set $\beta_r^{w,x}$ to be the number of $i \in [k]$ such that $\mathcal{A}_r^w(i)$ is determined (correctly) by $w\restriction_{A_r^0}$. We call $r \in \{0,1\}^\rho$ *typical* with respect to $\mathcal{A}$ if

$$\mathop{\mathbb{E}}_{w\sim\mathcal{D},x\sim\mathcal{D}}[\beta_r^{w,x}] \le \frac{2}{3}k\,.$$

Definition 4.5. For any typical r we call an $w \in \{0,1\}^n$ r-hard if

$$\mathop{\mathbb{E}}_{x\sim\mathcal{D}}[\beta_r^{w,x}] \le \frac{5}{6}k\,.$$

Lemma 4.6. *For any typical r,*

$$\Pr_{w\sim\mathcal{D}}[w \text{ is } r\text{-hard}] > \frac{4}{5}\,.$$

Proof. This proof follows directly from the definition of typical and r-hard and Markov's Inequality. $\square$

Lemma 4.7. *Let $\mathcal{A}$ be a message reconstructor for $\mathcal{C}$. Then* $\Pr_r$*[r is typical w. r. t. $\mathcal{A}$] $> 1/2$.*

We defer the proof of Lemma 4.7 to Section 4.1, and first show how to use it to prove Theorem 3.2.

For every random string $r \in \{0,1\}^\rho$ we denote by $N(r)$, $0 \le N(r) \le k$, the expected number of correctly reconstructed bits by $\mathcal{A}_r$, where the expectation is taken over $w \sim \mathcal{D}$. Formally,

$$N(r) = \mathop{\mathbb{E}}_{w\sim\mathcal{D}}\left[k - d(\mathcal{A}_r^w, \mathcal{C}^{-1}(D_{\mathcal{C}}(w)))\right].$$

Furthermore, given $y \in \mathcal{D}$ (by $y \in \mathcal{D}$ we mean $y \in \{0,1\}^n$ that is in the support of $\mathcal{D}$) and $S \subseteq [n]$ we denote by $N(r,y,S)$ the expected number of correctly reconstructed bits by $\mathcal{A}_r$, where the expectation is taken over $w \sim \mathcal{D}$ conditioned on the event $w\restriction_S = y\restriction_S$. Formally, $N(r,y,S)$ is equal to

$$\mathop{\mathbb{E}}_{w\sim\mathcal{D}}\left[k - d(\mathcal{A}_r^w, \mathcal{C}^{-1}(D_{\mathcal{C}}(w))) \;\middle|\; w\restriction_S = y\restriction_S\right].$$

Lemma 4.8. *For any typical r and y that is r-hard we have,*

$$N(r,y,A_r^0) \le k\left(1 - \frac{(\varepsilon/2)^{11q/10}}{3}\right).$$

Proof. By the definition of a typical r and the fact that y is r-hard, the expected number of bits whose reconstructed value either depends on non-influential indices or is guaranteed to be always wrong is at least $k/6$. Call these bits *free bits*. This means that every free bit may be incorrectly reconstructed by $\mathcal{A}_r$ for some assignment α to the non-influential indices (if it can be correctly reconstructed at all). So, by Fact 4.3 the probability that $\mathcal{A}_r$ fails on a specific free bit, taken over

$$w \sim \mathcal{D} \mid w\restriction_{A_r^0} = y\restriction_{A_r^0},$$

is at least $(\varepsilon/2)^{11q/10}$ (we can assume that the reconstructed value of each bit depends on at most $q = \log k \le \log n$ of the non-influential indices, since otherwise the query complexity is not as advertised and we are done and we know that $|A_r^0| < q/10$). Hence, by linearity of expectation, the expected number of bits that are reconstructed correctly by $\mathcal{A}_r$, taken over $w \sim \mathcal{D}$, is at most

$$\frac{5}{6}k + \frac{k}{6}\left(1 - \left(\frac{\varepsilon}{2}\right)^{11q/10}\right) = k\left(1 - \frac{(\varepsilon/2)^{11q/10}}{6}\right). \qquad \square$$

We now define a sequence of $T+1$ random variables forming a Doob martingale. For a fixed $r \in \{0,1\}^\rho$, $y \in \mathcal{D}$ and every t, $0 \le t \le T$, we define $H_t^{r,y} \triangleq N(r,y,A_r^0 \cup \cdots \cup A_r^t)$, that is the expected number of bits reconstructed correctly by $\mathcal{A}_r$, where the expectation is taken over all $w \sim \mathcal{D}$ that agree with y on all indices in $A_r^0 \cup A_r^1 \cup \cdots \cup A_r^t$. That is, in the tth step of the martingale we expose $y_{|A_r^t}$.

By Lemma 4.8,

$$H_0^{r,y} \le k\left(1 - \frac{(\varepsilon/2)^{11q/10}}{6}\right)$$

for all typical r and r-hard y. On the other hand, if $\mathcal{A}_r$ properly reconstructs y then $H_T^{r,y} = k$. Thus, for every typical r we have

$$\Pr_{y\sim\mathcal{D}}\left[\mathcal{A}_r^y = \mathcal{C}^{-1}(D_{\mathcal{C}}(y))\right] = \Pr_{y\sim\mathcal{D}}\left[H_T^{r,y} = k\right]$$

which is at most

$$\Pr_{y\sim\mathcal{D}}\left[H_T^{r,y} - H_0^{r,y} \ge k\frac{(\varepsilon/2)^{11q/10}}{6}\right].$$

By Lemma 4.2, $T \le k$ and the influence $I(A_r^t, r)$ of each set A_r^t, $1 \le t \le T$, is bounded by $10q$. Thus for all $1 \le t < T$ we have $|H_{t+1}^{r,y} - H_t^{r,y}| \le 10q$. Now we can apply Azuma's Inequality [2], by which we obtain that

$$\Pr_{y \sim \mathcal{D}}\left[H_T^{r,y} - H_0^{r,y} \ge k\frac{(\varepsilon/2)^{11q/10}}{6}\right]$$

is at most

$$\exp\left(\frac{-k^2(\varepsilon/2)^{11q/5}}{36T(10q)^2}\right) \le \exp\left(\frac{-k(\varepsilon/2)^{11q/5}}{3600q^2}\right)$$

where in the last inequality we used the fact that $T \le k$.

This means that the probability that $\mathcal{A}$ reconstructs all k bits correctly with a typical r is $o(1)$, unless $q = \Omega(\log k)$. Since a random r is typical with probability at least $1/2$ and y is r-hard with probability at least $4/5$, we conclude that unless $q = \Omega(\log k)$, $\mathcal{A}$ fails on $w \sim \mathcal{D}$ with overall probability at least $(1/2)(4/5) - o(1) = 2/5 - o(1)$. This completes the proof of Theorem 3.2, pending the proof of Lemma 4.7 which we present next.

4.1 Proof of Lemma 4.7

We need to prove that most random strings r satisfy

$$\mathop{\mathbb{E}}_{w,x \sim \mathcal{D}}[\beta_r^{w,x}] \le \frac{2}{3}k\,.$$

To this end, we need the following auxiliary lemma, which is essentially an entropy preservation argument.

Lemma 4.9. *For any two (possibly probabilistic) algorithms given by their functions, an encoder $E : \{0,1\}^k \to \{0,1\}^{k/10}$ and a decoder $D : \{0,1\}^{k/10} \to \{0,1\}^k$,*

$$\Pr_{r \sim \{0,1\}^\rho, x \sim \{0,1\}^k}\left[x = D(E(x))\right] \le 2^{-9k/10},$$

where r denotes the outcome of the random coin flips of E and D. Furthermore, the inequality holds even if E and D have shared randomness.

Proof. Let E, D be the two algorithms. Denote by E_r and D_r their deterministic versions operating with a fixed random seed r. For every possible r, partition the set $\{0,1\}^k$ into at most $m_r \le 2^{k/10}$ parts $X_1^r, \dots, X_{m_r}^r$ such for all i and $x, y \in X_i^r$ we have $E_r(x) = E_r(y)$. Observe that for every fixed r, and conditioned over the event that a random x falls in X_i^r, $\Pr_{x \sim X_i^r}[D_r(E_r(x)) = x] \le 1/|X_i^r|$, and clearly the probability that x falls in X_i^r is exactly $|X_i^r|/2^k$. So for every fixed r we have

$$\Pr_x\left[x = D_r(E_r(x))\right] = \sum_{i=1}^{m_r}\left(\frac{|X_i^r|}{2^k}\right)\left(\frac{1}{|X_i^r|}\right)$$

which is equal to $\sum_{i=1}^{m_r} 1/2^k \le 2^{-9k/10}$. Hence this holds for a random r as well. □

Now consider the following encoder/decoder pair E,D corresponding to $\mathcal{C}$ and $\mathcal{A}$. We will assume that E and D share a random string r, and describe their operation for every possible r.

The encoder E_r on $x \in \{0,1\}^k$ computes $y = \mathcal{C}(x)$, converts y into w by flipping each bit with probability $\varepsilon/2$ independently of the other bits, and outputs

$$E_r(x) \triangleq w\restriction_{A_r^0}$$

(the identity of the flipped bits is not part of the shared randomness). By the definition of A_r^0, $|E_r(x)| \leq k/10$ for every x and r. (If necessary, $E_r(x)$ can be padded arbitrarily up to length $k/10$.)

Before defining the decoder, let us denote by $S_r^z \subseteq [k]$ the set of bits for which the value is determined by $\mathcal{A}_r$, given that the assignment to the influential indices A_r^0 of the received word equals z.

The decoder D_r on $z \in \{0,1\}^{k/10}$ constructs

$$D_r(z) \triangleq x' \in \{0,1\}^k$$

by reconstructing x'_i according to $\mathcal{A}_r$ for all $i \in S_r^z$, and guessing $x'_i \in \{0,1\}$ uniformly at random for all other indices $i \in [k] \setminus S_r^z$.

We can now prove Lemma 4.7 by showing that the pair E,D defined above contradicts the statement of Lemma 4.9, unless for most random strings r,

$$\mathop{\mathbb{E}}_{w,x\sim\mathcal{D}}[\beta_r^{w,x}] \leq \frac{2}{3}k\,.$$

We start by observing that the distributions $E_r(x) : x \sim \{0,1\}^k$ and $w\restriction_{A_r^0} : w \sim \mathcal{D}$ are identical for every r. Therefore, since for every r the value $\beta_r^{w,x}$ depends only on $w\restriction_{A_r^0}$ and x, we have

$$\mathop{\mathbb{E}}_{w,x\sim\mathcal{D}}\left[\beta_r^{w,x}\right] = \mathop{\mathbb{E}}_{x\sim\{0,1\}^k}\left[\beta_r^{\mathrm{ext}(E_r(x)),x}\right]$$

for all r as well, where $\mathrm{ext}(E_r(x)) \in \{0,1\}^n$ is any arbitrary string (extension) whose restriction to A_r^0 equals $E_r(x)$.

Now assume towards a contradiction that for at least half of the random strings r,

$$\mathop{\mathbb{E}}_{w,x\sim\mathcal{D}}\left[\beta_r^{w,x}\right] = \mathop{\mathbb{E}}_{x\sim\{0,1\}^k}\left[\beta_r^{\mathrm{ext}(E_r(x)),x}\right] > \frac{2}{3}k\,.$$

Since $0 \leq \beta_r^{w,x} \leq k$ for all w, x and r, this means that for at least half of the random strings r,

$$\Pr_{x\in\{0,1\}^k}\left[\beta_r^{\mathrm{ext}(E_r(x)),x} \geq k/2\right] > \frac{1}{6}\,.$$

Therefore (taking into account that $\beta_r^{\mathrm{ext}(E_r(x)),x} > 0$ also implies that each of the bits not correctly determined by A_r^0 obtains the correct value with probability $1/2$ independently of the others),

$$\Pr_{r,x\sim\{0,1\}^k}\left[x = D_r(E_r(x))\right] \geq \left(\frac{1}{2}\right)\left(\frac{1}{6}\right)2^{-k/2}$$

which is more than $2^{-9k/10}$ and thus contradicting Lemma 4.9. □

5 Proof of Theorem 3.4

Outline: For the proof of Theorem 3.4 we show that there is an input distribution $\mathcal{D}$ on which any deterministic codeword reconstructor with query complexity $o(\sqrt{\log R(\mathcal{C})})$ fails with probability larger than $1/3$. This is done by constructing a set of indices $S \subset [n]$ such that for every deterministic reconstructor and for every $i \in S$, if E_i is the event that the reconstructor errs on index i, then the events E_i, $i \in S$ are independent (with respect to $\mathcal{D}$). Since the reconstructor fails unless it reconstructs all indices $i \in S$ correctly, the probability that it does not fail goes down exponentially with the size of the set S. Thus it is sufficient to show that the probability of each E_i is not too low, and that S is sufficiently large. The latter is done using the Sunflower Lemma [10] and the fact that $\mathcal{C}$ is a linear code.

Fix $\varepsilon > 0$. Let $\mathcal{C}$ be a family of linear $[n,k,d]$ codes and let $\mathcal{A}$ be its (q,ε) codeword reconstructor. Let $G_1,\ldots,G_n \in \{0,1\}^k$ denote the rows of $\mathcal{C}$'s generating matrix G, satisfying $\mathcal{C}(x)_i = \langle G_i, x\rangle \pmod 2$ for every $x \in \{0,1\}^k$ and $i \in [n]$, and let $\hat{n} \triangleq \mathrm{R}(\mathcal{C})$ denote the number of distinct rows in G.

The distribution $\mathcal{D}$ is defined similarly to the definition in Section 4, that is, a random word $w \sim \mathcal{D}$ is generated by picking a uniformly random $y \in \mathcal{C}$ and then obtaining w by flipping each of the bits of y with probability $\varepsilon/2$, independently of the other bits. As we also mentioned in Section 4, $w \sim \mathcal{D}$ is ε-close to $\mathcal{C}$ only with probability $1-o(1)$, but this will be sufficient because we will show that for any r, $\mathcal{A}_r$ will fail to reconstruct the correct codeword with probability at least $1/2$.

Lemma 5.1. *The following holds for $\mathcal{D}$ and any linear code $\mathcal{C}$:*

1. *Let $T \subseteq [n]$ and $i \in [n] \setminus T$ be such that G_i (the i-th row of $\mathcal{C}$'s generating matrix) is linearly independent of rows $\{G_j : j \in T\}$. Then for any $\alpha \in \{0,1\}^{|T|}$,*

$$\Pr_{w\sim\mathcal{D}:w\restriction_T=\alpha}\big[D_{\mathcal{C}}(w)_i = 1\big] \quad \textit{is equal to} \quad \Pr_{w\sim\mathcal{D}:w\restriction_T=\alpha}\big[D_{\mathcal{C}}(w)_i = 0\big]$$

   and thus both are equal to $1/2$, where "$w \sim \mathcal{D} : w\restriction_T = \alpha$" is shorthand for "$w \sim \mathcal{D}$ conditioned over the event that $w\restriction_T = \alpha$."

2. *Let $S_1,\ldots,S_t \subseteq [n]$ be disjoint sets with $|S_i| \le q \le \log n$ for all $i \in [t]$. For any sequence*

$$\langle \alpha_i \in \{0,1\}^{|S_i|}\rangle_{i\in[t]}$$

   of partial assignments, we have that

$$\Pr_{w\sim\mathcal{D}}\big[w\restriction_{S_i} = \alpha_i \text{ for some } i \in [t]\big] \quad \textit{is at least} \quad 1-\big(1-(\varepsilon/2)^q\big)^t.$$

Proof. For the first item, notice that since the codewords of $\mathcal{C}$ form a linear subspace, for every $i \in [n]$ we have $\Pr_{y\in\mathcal{C}}[y_i = 1] = \Pr_{y\in\mathcal{C}}[y_i = 0] = 1/2$ (here we assume without loss of generality that $\mathcal{C}$ is not redundant, i. e., the generating matrix of $\mathcal{C}$ has no all-zero rows). Conditioning the above over some restriction to $y\restriction_T$ has no effect on indices i that are linearly independent of the indices in T. This holds since the subset of $\mathcal{C}$ formed by the restriction is an affine subspace not parallel to the kernel of G_i. Finally, in $\mathcal{D}$ the i-th bit of y can be flipped with some probability, but this has no effect since the probability of y_i being flipped is independent of the value y_i.

The second item follows from immediately from the definition of $\mathcal{D}$. □

Recall that $Q_{i,r} \subseteq [n]$ is the set of indices queried by $\mathcal{A}$ on input $i \in [n]$ and random string r. From this point on let us fix r and prove that unless

$$q \triangleq \max_{i \in [n]} \left\{ |Q_{i,r}| \right\}$$

is of order $\Omega(\sqrt{\log \hat{n}})$, the probability (over $w \sim \mathcal{D}$) that $\mathcal{A}_r$ correctly reconstructs w (into $y = D_{\mathcal{C}}(w)$) is less than $1/2$. Since r is fixed, we will make the notation shorter by omitting the subscript r from the sets $Q_{i,r}$.

The proof proceeds in two steps. In the first step we find a large subset $F \subseteq [n]$ of indices, such that $Q_i \cap Q_j = Q_{i'} \cap Q_{j'}$ for all $i, j, i', j' \in F$, and in addition, for all $i, j \in F$ the values $D_{\mathcal{C}}(w)_i, D_{\mathcal{C}}(w)_j$ are independent of

$$w\restriction_{Q_i \cap Q_j}.$$

F is constructed by first finding a large sunflower in the sets $Q_1, \ldots, Q_n$ and then removing from it the (not too many) "bad" petals that violate the above property. In the second part of the proof we show that given a large enough set F as above, the probability that $\mathcal{A}_r$ incorrectly reconstructs at least one of the indices in F is high.

Definition 5.2. A *sunflower* with t *petals* and a *core* T is a collection of sets, $Q_1, \ldots, Q_t$, such that $Q_i \cap Q_j = T$ for all $i \neq j$. We call the number of petals the *size* of the sunflower.

Lemma 5.3 (Sunflower Lemma [10])**.** *Let $\mathcal{Q}$ be a family of n sets, each set having cardinality at most q. If $n > q!(t-1)^q$ then $\mathcal{Q}$ contains a sunflower with t petals. In particular, $\mathcal{Q}$ contains a sunflower of size at least $(1/q)n^{1/q}$.*

Let R be a set of $\hat{n}$ indices corresponding to $\hat{n}$ distinct rows in G. Let $\mathcal{Q} = \langle Q_i \rangle_{i \in R}$ be the family of sets queried by $\mathcal{A}_r$ on inputs $i \in R$. By definition, $\mathcal{Q}$ contains $\hat{n}$ sets, each of size at most q. From Lemma 5.3 we can obtain a sunflower $\mathcal{S} \subseteq \mathcal{Q}$, $\mathcal{S} = Q_{i_1}, \ldots, Q_{i_t}$, with $t \geq (1/q)\hat{n}^{1/q}$ petals. Let $T \subset [n]$ denote the core of $\mathcal{S}$. We define the *span* of the core T as $\operatorname{span}(T) = \operatorname{span}\{G_i : i \in T\}$ which is equal to

$$\left\{ \sum_{i \in T} \alpha_i G_i \pmod 2 : \forall i, \alpha_i \in \{0,1\} \right\}$$

which is a subset of $\{0,1\}^k$. Since $|T| \leq q$ the span of T contains at most 2^q different rows from G.

Next we form a family $\mathcal{S}' \subseteq \mathcal{S}$ of sets by removing from $\mathcal{S}$ every petal Q_{i_j} for which G_{i_j} belongs to the span of T. Namely, we set

$$\mathcal{S}' \triangleq \left\{ Q_{i_j} \in \mathcal{S} : G_{i_j} \notin \operatorname{span}(T) \right\}.$$

Notice that the resulting family $\mathcal{S}'$ is a sunflower as well, with the same core T. Furthermore, the size of $\mathcal{S}'$ is at least $t' \geq (1/q)\hat{n}^{1/q} - 2^q$.

Intuitively, the query sets in $\mathcal{S}'$ correspond to those indices in R that are "independent" of $w\restriction_T$. We call these indices *free indices* and denote their set by F. Namely, $F = \{i : Q_i \in \mathcal{S}'\}$. By the first item of Lemma 5.1, for every free index $i \in F$ and all $\alpha \in \{0,1\}^{|T|}$ we have that $\Pr_{w \sim \mathcal{D} : w\restriction_T = \alpha}[D_{\mathcal{C}}(w)_i = 1]$ is same as $\Pr_{w \sim \mathcal{D} : w\restriction_T = \alpha}[D_{\mathcal{C}}(w)_i = 0]$:

$$\Pr_{w \sim \mathcal{D} : w\restriction_T = \alpha}[D_{\mathcal{C}}(w)_i = 1] \;=\; \Pr_{w \sim \mathcal{D} : w\restriction_T = \alpha}[D_{\mathcal{C}}(w)_i = 0] \;=\; \frac{1}{2}. \tag{5.1}$$

Now we can show that if $q = o(\sqrt{\log \hat{n}})$, then with probability at least $1/2$ one of the free indices will be reconstructed incorrectly by $\mathcal{A}_r$. Assume that the contrary is true, i. e., $\Pr_{w\sim\mathcal{D}}[\beta_w] > 1/2$, where β_w is the indicator of the event that $\mathcal{A}_r$ reconstructs all free indices correctly. This implies that there exists an $\alpha \in \{0,1\}^{|T|}$ such that $\Pr_{w\sim\mathcal{D}:w|_T=\alpha}[\beta_w] > 1/2$. If for some $i \in F$ the value of $\mathcal{A}_r^w(i)$ is determined by the fact that $w\restriction_T = \alpha$ then by equation (5.1) the probability that $\mathcal{A}_r$ reconstructs i incorrectly is $1/2$. So it must be the case that having $w\restriction_T = \alpha$ does not determine $\mathcal{A}_r^w(i)$ for any of the free indices i, and in particular, for every $S_i \triangleq Q_i \setminus T$, $Q_i \in \mathcal{S}'$ there must be an assignment α_i to $w\restriction_{S_i}$ for which $\mathcal{A}_r$ reconstructs i incorrectly. Since the sets S_i are disjoint we can apply the second item of Lemma 5.1 by which the probability that one of the free indices is reconstructed incorrectly is at least

$$1-(1-(\varepsilon/2)^q)^{|\mathcal{S}'|} \geq 1-(1-(\varepsilon/2)^q)^{\frac{1}{q}\hat{n}^{1/q}-2^q}$$

which is at least

$$1-\mathrm{e}^{-(\varepsilon/2)^q(\frac{1}{q}\hat{n}^{1/q}-2^q)}\,.$$

Notice that the above probability is larger than $1/2$ (in fact it is almost 1) if

$$(\varepsilon/2)^q\left(\frac{1}{q}\hat{n}^{1/q}-2^q\right) > 10\,,$$

which is the case for $q = o(\sqrt{\log \hat{n}})$. Hence a contradiction. $\square$

6 Extensions and open problems

6.1 Reconstruction against random noise

In the usual definition of locally decodable codes (and self-reconstructable codes) it is assumed that the noise is adversarial, i. e., that the set of corrupted locations is chosen in a worst case manner. Our definition of message and codeword reconstruction is against adversarial noise as well. But does reconstruction become easier in the random-noise model, where the received word w is obtained by flipping (independently) each bit of the codeword y with probability at most ε, and the requirement is that for every $y \in \mathcal{C}$ the decoding/correction/reconstruction is successful with probability at least $2/3$, taken over both r and w?

While both local decoding and self-correction become trivial in the random-noise model (via simple repetition codes), this is not the case for reconstruction. In fact, our proofs used input distributions that exactly correspond to the random noise model. Moreover, there is a general reduction that allows to translate any query-complexity lower bound in the adversarial-noise model for message reconstruction to a lower bound in the random-noise model (or alternatively, translate any upper bound in the random-noise message reconstruction model into one that works in the adversarial-noise model) via LDCs:

Claim 6.1. *Let $\mathcal{C}$ be a code with a q-query message reconstructor against random noise, and let H be a p-query LDC. Then the code $H(\mathcal{C})(x) \triangleq H(\mathcal{C}(x))$ (the LDC H composed with $\mathcal{C}$) has an $O(pq)$-query message reconstructor in the adversarial-noise model.*

We omit the formal proof of Claim 6.1, but the main idea is that the additional LDC encoding enables converting adversarial noise into random noise. This is the case since by the definition of an LDC, every bit of the codeword $y \in \mathcal{C}$ can be decoded correctly with high probability, independently of other bits.

Combining Claim 6.1 with Theorem 3.2, and using the fact that there are constant query LDCs, we get that message reconstruction against random noise requires $\Omega(\log k)$ queries. We can also deduce correlations for lower bounds concerning codeword reconstruction, however here the size of the LDC used becomes important as it affects the codeword size of the combined code.

We note that while there are codes of super-polynomial length with an $O(\log k)$-query message reconstructor, we do not know any polynomial-length code with an $O(\log^2 k)$-query message reconstructor. So, the real bound in Theorem 3.2 may be higher than $\Omega(\log k)$ for efficient codes. On the other hand, the repetition code, whose encoding is obtained by concatenating $O(\log k)$ copies of the original message, has a trivial $\log k$-query message reconstructor against random noise. Thus for random noise our lower bound is optimal, irrespective of the length of the code.

6.2 Partial reconstruction

In a more general setting, we may require that a message reconstructor should decode correctly any specified t bits of the message, instead of all k. It is straightforward to extend the lower bound in Theorem 3.2 for this generalization to $\Omega(\log t)$. Similarly, if we require that a codeword reconstructor should decode correctly any t bits of the codeword instead of all n, then we can extend Theorem 3.4 to provide a lower bound of $\Omega(\sqrt{\log \hat{t}})$, where $\hat{t}$ is the number of distinct rows in the t rows corresponding to the decoded part.

6.3 Open problems

- Our results suggest that one cannot get message reconstructors that are significantly more query-efficient than the amplified versions of constant-query LDCs. It is an interesting question whether this connection is bidirectional, i. e., whether q-query message reconstruction implies q-query local decoding with error probability $O(1/k)$.

- Are there codes of *polynomial* length that have $O(\log k)$-query message reconstructors? A positive answer would be an intermediate step towards proving that efficient constant query LDCs exist.

- In the case of codeword reconstruction, Theorem 3.4 says that for any linear code the codeword reconstructor must have query complexity $\Omega(\sqrt{\log \hat{n}})$. On the other hand we have linear codes with $O(\log \hat{n})$-query codeword reconstructor. We expect the upper bound to be the correct one, but we are currently unable to close this gap.

Acknowledgments

We thank C. Seshadhri, David Garcia-Soreano and Ronald de Wolf for useful discussions on the topic. In particular, we thank Ronald de Wolf for many valuable comments on an early draft of this paper.

References

[1] NIR AILON, BERNARD CHAZELLE, SESHADHRI COMANDUR, AND DING LIU: Property-preserving data reconstruction. *Algorithmica*, 51(2):160–182, 2008. Preliminary version in ISAAC'04. [doi:10.1007/s00453-007-9075-9] 517

[2] NOGA ALON AND JOEL H. SPENCER: *The Probabilistic Method*. Wiley, 1992. 525

[3] LÁSZLÓ BABAI, LANCE FORTNOW, LEONID A. LEVIN, AND MARIO SZEGEDY: Checking computations in polylogarithmic time. In *Proc. 23rd STOC*, pp. 21–31. ACM Press, 1991. [doi:10.1145/103418.103428] 518

[4] ARNAB BHATTACHARYYA, ELENA GRIGORESCU, MADHAV JHA, KYOMIN JUNG, SOFYA RASKHODNIKOVA, AND DAVID P. WOODRUFF: Lower bounds for local monotonicity reconstruction from transitive-closure spanners. *SIAM J. Discrete Math.*, 26(2):618–646, 2012. Preliminary version in RANDOM'10. [doi:10.1137/100808186] 517

[5] MANUEL BLUM, MICHAEL LUBY, AND RONITT RUBINFELD: Self-testing/correcting with applications to numerical problems. *J. Comput. System Sci.*, 47(3):549–595, 1993. Preliminary version in STOC'90. [doi:10.1016/0022-0000(93)90044-W] 517

[6] SOURAV CHAKRABORTY, ELDAR FISCHER, AND ARIE MATSLIAH: Query complexity lower bounds for reconstruction of codes. In *Proc. 2nd Symp. on Innovations in Computer Science (ICS'11)*, pp. 264–274, 2011. Accessible at ICS. See also ECCC. 515

[7] BERNARD CHAZELLE AND C. SESHADHRI: Online geometric reconstruction. *J. ACM*, 58(4):14, 2011. Preliminary version in SoCG'06. [doi:10.1145/1989727.1989728] 517

[8] BENNY CHOR, ODED GOLDREICH, EYAL KUSHILEVITZ, AND MADHU SUDAN: Private information retrieval. *J. ACM*, 45(6):965–981, 1998. Preliminary version in FOCS'95. [doi:10.1145/293347.293350] 518

[9] KLIM EFREMENKO: 3-query locally decodable codes of subexponential length. *SIAM J. Comput.*, 41(6):1694–1703, 2012. Preliminary version in STOC'09. [doi:10.1137/090772721] 518

[10] PAUL ERDŐS AND RICHARD RADO: Intersection theorems for systems of sets. *J. London Math. Soc.*, 35:85–90, 1960. Accessible at Alfréd Rényi Institute of Mathematics. 527, 528

[11] WILLIAM I. GASARCH: A survey on private information retrieval (computational complexity column). *Bulletin of the EATCS*, 82:72–107, 2004. Accessible at Author's Website. 518

[12] ODED GOLDREICH, SHAFI GOLDWASSER, AND DANA RON: Property testing and its connection to learning and approximation. *J. ACM*, 45(4):653–750, 1998. Preliminary version in FOCS'96. [doi:10.1145/285055.285060] 517, 518

[13] MADHAV JHA AND SOFYA RASKHODNIKOVA: Testing and reconstruction of Lipschitz functions with applications to data privacy. *SIAM J. Comput.*, 42(2):700–731, 2013. Preliminary version in FOCS'11. See also ECCC. [doi:10.1137/110840741] 517

[14] SATYEN KALE, YUVAL PERES, AND C. SESHADHRI: Noise tolerance of expanders and sublinear expander reconstruction. *SIAM J. Comput.*, 42(1):305–323. Preliminary version in FOCS'08. [doi:10.1137/110837863] 517

[15] JONATHAN KATZ AND LUCA TREVISAN: On the efficiency of local decoding procedures for error-correcting codes. In *Proc. 32nd STOC*, pp. 80–86. ACM Press, 2000. [doi:10.1145/335305.335315] 516, 518

[16] IORDANIS KERENIDIS AND RONALD DE WOLF: Exponential lower bound for 2-query locally decodable codes via a quantum argument. *J. Comput. System Sci.*, 69(3):395–420, 2004. Preliminary versions in STOC'03 and CoRR. [doi:10.1016/j.jcss.2004.04.007, arXiv:quant-ph/0208062] 518

[17] MICHAEL E. SAKS AND C. SESHADHRI: Parallel monotonicity reconstruction. In *Proc. 19th Ann. ACM-SIAM Symp. on Discrete Algorithms (SODA'08)*, pp. 962–971. ACM Press, 2008. Accessible at ACM DL. 517

[18] LUCA TREVISAN: Some applications of coding theory in computational complexity. *Quaderni di Matematica*, 13:347–424, 2004. See ECCC and CoRR. [arXiv:cs/0409044] 518

[19] STEPHANIE WEHNER AND RONALD DE WOLF: Improved lower bounds for locally decodable codes and private information retrieval. In *Proc. 32th Internat. Colloq. on Automata, Languages and Programming (ICALP'05)*, pp. 1424–1436, 2005. [doi:10.1007/11523468_115, arXiv:quant-ph/0403140] 518

[20] DAVID P. WOODRUFF: New lower bounds for general locally decodable codes. *Electronic Colloquium on Computational Complexity (ECCC)*, 14(006), 2007. Accessible at ECCC. 518

[21] SERGEY YEKHANIN: Towards 3-query locally decodable codes of subexponential length. *J. ACM*, 55(1):1, 2008. Preliminary version in STOC'07. [doi:10.1145/1326554.1326555] 518

AUTHORS

Sourav Chakraborty
Associate professor
Chennai Mathematical Institute, Chennai, Inda
sourav@cmi.ac.in
http://www.cmi.ac.in/~sourav

Eldar Fischer
Faculty of Computer Science
Technion – Israel Institute of Technology
Technion City, Haifa 32000, Israel
eldar@cs.technion.ac.il
http://www.cs.technion.ac.il/users/eldar

Arie Matsliah
Software Engineer
Google, Mountain View, CA
ariem@google.com

ABOUT THE AUTHORS

SOURAV CHAKRABORTY is an associate professor at the Chennai Mathematical Institute, India. He completed his Ph. D. in 2008 at the University of Chicago under the supervision of László Babai. His research interests lie in the classical and quantum complexity of Boolean functions (including property testing and different combinatorial measures of the complexity of Boolean functions), electronic commerce, graph algorithms, and coding theory.

ELDAR FISCHER completed his Ph. D. in 1999 at Tel-Aviv University under the supervision of Noga Alon, and has been a member of the Faculty of Computer Science at the Israel Institute of Technology (the Technion) since 2001. His main interests lie in the analysis of algorithms, especially property testers and sublinear-time algorithms, but he is also interested in most things combinatorial. His main extracurricular interest centers on board and card games.

ARIE MATSLIAH is a software engineer in the ads-quality team at Google. He graduated from the Technion – Israel Institute of Technology in 2008; his advisor was Eldar Fischer. His research interests include sublinear algorithms and complexity of Boolean functions.