Better Lower Bounds for Locally Decodable Codes

Amit Deshpande* Rahul Jain† T Kavitha† Satyanarayana V. Lokam‡
Jaikumar Radhakrishnan†

November 12, 2001

Abstract

An error-correcting code is said to be *locally decodable* if a randomized algorithm can recover any single bit of a message by reading only a small number of symbols of a possibly corrupted encoding of the message.

Katz and Trevisan [KT] showed that any such code $C : \{0,1\} \to \Sigma^m$ with a decoding algorithm that makes at most q probes must satisfy $m = \Omega((n/\log|\Sigma|)^{q/(q-1)})$. They assumed that the decoding algorithm is non-adaptive, and left open the question of proving similar bounds for adaptive decoders.

We improve the results of [KT] in two ways. First, we give a simpler proof of their result. Second, and this is our main result, we prove that $m = \Omega((n/\log|\Sigma|)^{q/(q-1)})$ even if the decoding algorithm is adaptive. An important byproduct of our proofs is a randomized method for *smoothening* the adaptive decoding algorithm.

1 Introduction

1.1 Definition of the problem and background:

Let $C : \{0,1\}^n \to \Sigma^m$. We say that C is a (q, δ, ϵ)-*locally decodable code* if there is a randomized algorithm that can recover, with probability at least $\frac{1}{2} + \epsilon$, any bit of a message x by reading a small number of symbols from code word $C(x)$, even if δm entries of $C(x)$ are corrupted. Locally decodable codes arise in several contexts in computational complexity and cryptography: self-correcting computations [GLRSW, GS], probabilistically checkable proofs [BFLS], conversion of worst-case hardness to average-case hardness in the construction of pseudo-random generators [BFNW], and private information retrieval [CGKS, BI, Ma]. These codes could also have potential applications in fault-tolerant data storage [KT] .

An important question about a locally decodable code concerns the trade-off between the length of a codeword m and the number of queries q. In the construction of traditional error-correcting codes, an important goal is to have a linear time decoding algorithm while achieving $m = O(n)$ and tolerating upto a constant fraction δ of errors in the received word (possibly corrupted encoding). It is amazing that there are error-correcting codes fulfilling all these requirements. Note that in this

*Chennai Mathematical Institute, Chennai, India. E-Mail: amit@cmi.ac.in

†Theoretical Computer Science Group, Tata Institute of Fundamental Research (TIFR), Mumbai, India. E-Mail: {rahulj,kavita,jaikumar}@tcs.tifr.res.in

‡*Corresponding Author.* E-Mail: satyalv@eecs.umich.edu, Address: Department of Electrical Engineering and Computer Science, University of Michigan, 1301 Beal Avenue – EECS, Ann Arbor, MI 48109-2122. Part of the work done while visiting TIFR. Supported in part by NSF grant CCR-9988359

standard setting, we typically recover the *entire message* from the received word. Thus a linear time decoding procedure has a constant *amortized* time per message bit recovered. A natural question, then, is to ask if similar performance bounds can be achieved for a locally decodable code. Ideally, one would like to construct a locally decodable code with $m = O(n)$ that allows the recovery of any *single bit* of a message in randomized *constant time* from a corrupted encoding with up to a constant fraction of errors. Known constructions of locally decodable codes are far from achieving this goal. In the context of probabilistically checkable proofs, [BFLS] construct a locally decodable code with $m = n^{1+\epsilon}$ and $q = (\log n)^{O(1/\epsilon)}$. Based on results on private information retrieval [Am, G, BI], it is possible to construct a locally decodable code with $m = 2^{O(n^{1/(q-1)})}$ for constant q.

This paper is concerned with proving lower bounds for locally decodable codes. The first lower bound on the size of locally decodable codes was proved by Katz and Trevisan [KT], who showed for a *special class of decodable codes*,

$$m = \Omega\left(\left(\frac{\delta}{q^2}\right)^{\frac{1}{q-1}} \epsilon^{\frac{2(q+1)}{q-1}} \left(\frac{n}{\log|\Sigma|}\right)^{\frac{q}{q-1}}\right). \tag{1}$$

Thus, for such codes it is impossible to achieve $m = O(n)$ and $q = O(1)$ simultaneously if $\log|\Sigma| \ll n^{\frac{1}{q}}$. This bound, however, holds only for codes where the decoding algorithm is non-adaptive (that is, it decides on which locations to probe based just on its random string). In support of this restriction on decoders, Katz and Trevisan [KT] observe that all known constructions for locally decodable codes use non-adaptive decoders. They, however, state the following:

> One open questions 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.

In this paper we address this question, and prove lower bounds for locally decodable code without assuming that the decoder is non-adaptive, that is, we allow it to use what it reads in one probe to decide where to make the next probe. Katz and Trevisan observe that a general (q, δ, ϵ)-locally decodable code is also a $(\frac{|\Sigma|^q - 1}{|\Sigma| - 1}, \delta, \epsilon)$-locally decodable code and a $(q, \delta, \epsilon/|\Sigma|^{q-1})$-locally decodable code with *non-adaptive* decoders, and non-trivial lower bounds for general codes can be derived from (1). The bounds derived via such reductions deteriorate rapidly with $|\Sigma|$; in particular, they are not superlinear when $|\Sigma|$ is n (i.e. word size is a $\log n$). In some applications, such as private information retrieval, locally decodable codes with large alphabets are preferred. Hence, it is desirable to have a lower bound that deteriorates slowly as $|\Sigma|$ increases.

1.2 Our contributions

We will abuse terminology a little and use *adaptive codes* to refer to locally decodable codes with adaptive decoders, and *non-adaptive codes* to refer to locally decodable codes with non-adaptive decoders.

As an intermediate step in their proof of (1), Katz and Trevisan considered codes with decoding algorithms that probe the codeword almost uniformly: say no location is probed with probability more than c/m. Using information theoretic arguments they then showed that for such codes

$$m = \Omega\left(\left(\frac{1}{qc}\right)^{\frac{1}{q-1}} \epsilon^{\frac{2(q+1)}{q-1}} \left(\frac{n}{\log|\Sigma|}\right)^{\frac{q}{q-1}}\right). \tag{2}$$

We prove the following.

Theorem 1 *If the decoder probes no location with probability more that c/m, then*

$$m = \Omega\left(\left(\frac{\epsilon^2}{qc}\right)^{\frac{1}{q-1}}\left(\frac{n}{\log|\Sigma|}\right)^{\frac{q}{q-1}}\right).$$

This improves (2) in three ways: the bound is slightly better, the proof is simpler, and above all, it works for adaptive codes.

As stated above, the lower bounds for general non-adaptive codes follow from lower bounds for smooth codes. Although there does not appear to be any such reduction from general adaptive codes to general smooth codes, we obtain similar improvements for general non-adaptive codes. (See below for an overview of our technique.)

Theorem 2 *For any (q, δ, ϵ)-locally decodable code $C : \{0,1\}^n \to \Sigma$,*

$$m = \Omega\left(\epsilon^{\frac{q+1}{q-1}}\left(\frac{1}{q}\right)^{\frac{3q-1}{q-1}}\delta^{\frac{1}{q-1}}\left(\frac{n}{\log|\Sigma|}\right)^{\frac{q}{q-1}}\right).$$

In particular, this generalizes (1), for it shows that it is impossible to achieve $m = O(n)$, $q = O(1)$ and $\log|\Sigma| \ll n^{\frac{1}{q}}$ simultaneously, even with adaptive decoders.

We conclude that constant-time recovery of a single message bit is impossible for a linear length locally decodable code, whether or not the decoder is adaptive. In fact, from Theorem 2, we see that any such decoder must examine at least $\Omega(\log n/\log\log n)$ bits of the codeword.

1.3 Techniques

The proof in Katz and Trevisan [KT] was based on an information theoretic idea (used earlier by Mann [Ma] to prove lower bounds for private information retrieval):

> *a locally decodable code must repeat information about each bit in different parts of the codeword, so it must be possible to recover the string x by reading a small fraction (tending to 0 as n tends to infinity) of $C(x)$. To recover x one must read $\Omega(n)$ bits, so even this vanishingly small fraction must contain $\Omega(n/\log|\Sigma|)$ symbols, hence $|C(x)| = \omega(n/\log|\Sigma|)$.*

Our proofs also uses this idea, which we formalize as follows.

Lemma 1.1 *Let $C : \{0,1\}^n \to \Sigma^m$ and $\mathcal{A} : [n]\times\Sigma^*\times 2^{[m]} \to \{0,1\}$. Suppose S is a random variable taking values as subsets of $[m]$ such that $\forall i \in [n]\ \forall x \in \{0,1\}^n\ \ \Pr[\mathcal{A}(i, C(x)[S], S) = x_i] \geq \frac{2}{3}$, where $C(x)[S]$ is the substring of $C(x)$ corresponding to indices in S. Then, $\mathbf{E}[|S|]\cdot\log_2|\Sigma| \geq n(1-H(\frac{2}{3}))$.*

(The proof is in the appendix.) To use this lemma, we need to design $\mathcal{A}$, which can recover x from a small sample of symbols of $C(x)$. All our proofs use the following **generic method**: pick a sample of locations S, by picking each $a \in [m]$ independently with probability p; then, to recover x_i use the q-probe decoder that comes with the locally decodable code, run it on all random strings, discard runs that need to probe locations outside S, and return the answer given by the majority of the remaining runs.

Smooth/non-adaptive codes: Most of the work is in proving that the generic method guesses x_i correctly with probability at least $\frac{2}{3}$. It is relatively straightforward to prove that the runs of the decoder that successfully terminate within S give, on an average, significantly more correct answers than wrong answers. The problem is in showing that they give more correct answers than wrong answers *with high probability.* This is not hard to prove if the code has a *smooth* decoder, which does not probe any location with probability more than c/m. In fact, the random variable 'number of correct answers minus the number of wrong answers' can be expressed as a sum of $\{+1, -1\}$ random variables. Now, it is well known that in such situations, if the pair-wise dependency among the variables is small, then the variance of their sum is small. The *smoothness* condition translated to our setting tells us precisely this—the random variables corresponding to different runs of the decoder have few pair-wise dependencies.

Katz and Trevisan [KT] also used a decoding method based on sampling. However, instead of estimating the variance of the associated random variable, they identified a set of good runs of the decoder and, using the assumption that the decoder is non-adaptive and smooth, forced independence among them. Since we don't try to force independence, we are able to work even when the decoder is adaptive. Also, their method succeeded with probability only $\frac{1}{2} + \frac{\epsilon}{2}$ instead of $\frac{2}{3}$. (This difference gets further amplified when one tries to derive lower bounds for adaptive codes from the lower bounds for non-adaptive codes.)

To prove lower bounds for (not necessarily smooth) non-adaptive codes Katz and Trevisan [KT] showed that a (q, δ, ϵ)-locally decodable code can always be made smooth, with $c = q/\delta$. Using this reduction, we can take $c = q/\delta$ in Theorem 1, and slightly improve the previous best bound (2) for non-adaptive codes.

Adaptive Codes: Theorem 1 gives lower bounds for smooth adaptive codes. As stated above, the smoothness condition was inspired by its connection to general non-adaptive codes. Unfortunately, there does not appear to be such a simple connection when the decoder is adaptive. To describe this difficulty, we need to recall how non-adaptive codes are converted into smooth codes. When the decoder is non-adaptive, we know the pattern of probes it is going to make, even without looking at the codeword. In particular, we can identify the set of locations, Z, that are proved with probability more than $q/(\delta m)$. Since the decoder reads at most q entries in each invocation, the number of such locations is less than δm. Modify the decoding algorithm so that instead of probing a locations in Z explicitly, it assumes that the symbol 0 is stored there. The decoder is now "smoothened", and Since $|Z| \leq \delta m$, the new decoder continues to give the correct answer with probability at least $\frac{1}{2} + \epsilon$.

This argument fails for adaptive decoder, for now one cannot, without reading the codeword, determine which locations are heavily probed. In fact, different locations might be heavily probed for different codewords. There is another more subtle difficulty. Even if we are given the set of locations that are probed heavily, we cannot just go ahead and assume that they contain a 0, because this might change the pattern of accesses to $C(x)$, and some other locations may become heavily probed! It is, thus, not clear how a *smoothener* can be fixed in advance. We get around these difficulties using two main ideas: (i) we observe that to bound the variance in our sampling method it is enough that a weaker requirement than the smoothness condition above holds, and define a notion of Δ-smoothness to capture this; (ii) we show how a Δ-smoothener can be constructed by randomly sampling the codeword.

For want of space, we will not present the construction of Δ-smoothener in this abstract (the details are in the appendix). So, we will now informally describe here the main ideas surrounding a Δ-smoothener and its randomized construction.

Fix $i \in [n]$ and think of the randomized algorithm that predicts x_i as a probability distribution on decision trees. Each decision tree has depth at most q and every internal node is labelled by an address that the algorithm is supposed to probe when the control reaches that node. Each leaf contains the answer given by the decoder when it follows the corresponding path by probing the codeword. Fixing the codeword, we get a set of paths from the trees corresponding to different random choices of the decoder. We would like to determine locations that are probed very often in these paths, and smoothen the decoder by assuming that they contain a 0. Unfortunately, one cannot identify these locations precisely by reading just a few bits of the input. So, we make do with an approximation. There are two interesting features in our approximation. First, the notion of 'highly probed location' is now more nuanced—we impose different thresholds for different probes. For example, for the first probe we require that no location is probed more than Δ times, whereas for the second probe we tolerate locations that are probed about Δ/p times. Second, our method for constructing the Δ-smoothener is itself randomized—we 'learn' a smoothener by probing the codeword. The use of randomness, inevitably, comes with the risk of making errors. It is possible that there are some highly probed locations that are missed out by our Δ-smoothener. To compensate for this, we will allow a small number of trees to set aside, so that with respect to the rest all our thresholds are respected. The final product is is a set of positions $Z \subseteq [m]$ of the codeword such that, excluding a small set Bad of paths, all positions outside Z are probed with not too high a probability.

We construct a Δ-smoothener iteratively picking a set Z_k in step k, for $1 \leq k \leq q$, where Z_k is supposed to smoothen the kth probe. Since the first probe is non-adaptive, Z_1 can be determined without reading the codeword. In general, having determined $Z_1, Z_2, \ldots, Z_{k-1}$, we determine Z_k by picking a small random sample of locations from $C(x)$ and using them to detect the set of highly probed locations, Z_k, for the kth probe. The set $Z := \bigcup_{k=1}^{q} Z_k$ will be our Δ-smoothener. The technical part of our proof (presented in the appendix) shows that if we assume that all locations in Z contain a 0, we set aside a small number of random choices of the decoder, the resulting algorithm will, with high probability, be Δ-smooth. Also, $|Z| \leq \delta m$.

With such a Δ-smoothener Z at hand, we can use the sampling method to predict any x_i with high probability. Note, however, that in applying Lemma 1.1, the set S now includes not only the random bits picked to guess x_i, but also those used to generate the Δ-smoothener Z.

Organization of the paper: In Section 2, we consider smooth codes and prove Theorem 1. In Section 3, we define Δ-smootheners, and assuming that they can be constructed efficiently, complete the proof of Theorem 2. In the appendix, we give the proof of the information theoretic Lemma 1.1 described above, and show that Δ-smootheners can be constructed efficiently.

2 Smooth Codes

Notation: We assume that the randomized decoding algorithm uses a random string r, which is uniformly distributed in the set $[t] = \{1, 2, \ldots, t\}$. We will assume that $t \geq 10m$. (We can always ensure this by making k copies of each decision tree, for some large enough k.)

Definition 2.1 *Fix $x \in \{0,1\}^n$ and $i \in [n]$ and consider the decoding algorithm that predicts x_i by making probes to $C(x)$. For $r \in [t]$, let*

- *$e_r(x, i)$ be the set of locations in $[m]$ that the decoding algorithm probes for predicting x_i when the random string is r;*

- $ans_r(x,i) \in \{0,1\}$ *be the answer returned by the decoding algorithm, when the random string is* r*;*
- $adv_r(x,i) = \begin{cases} +1 & \text{if } ans_r(x,i) = x_i \\ -1 & \text{if } ans_r(x,i) \neq x_i \end{cases}$.

When x *and* i *are clear from the context, we drop them from our notation, and write* e_r*,* ans_r*, and* adv_r *respcetively.*

Smoothness assumption: The decoding algorithm reads no entry of the codeword with probability more than c/m. That is, for each $j \in [m]$,

$$\deg j \stackrel{\text{def}}{=} |\{r : j \in e_r(x,i)\}| \leq \left(\frac{c}{m}\right) t. \tag{3}$$

We will use Lemma 1.1 to show a lower bound for smooth codes.

Sampling: Let S be a random subset of $[m]$ obtained by picking each $j \in [m]$ independently with probability p (to be fixed later).

Prediction Algorithm $\mathcal{A}$: Fix x and i. For $r \in [t]$, let χ_r be the 0-1 random variable defined by

$$\Pr[\chi_r = 1 \mid S = \hat{S}] = \begin{cases} 0 & \text{if } e_r \not\subseteq \hat{S} \\ p^{q-|e_r|} & \text{if } e_r \subseteq \hat{S} \end{cases}.$$

(The additional random choices required in the case '$e_r \subseteq \hat{S}$' are made independently for different r.) Note that $\Pr[\chi_r = 1] = p^q$ for all $r \in [t]$. Now, let

$$\mathcal{A}(i, C(x)[S], S) \stackrel{\text{def}}{=} \underset{r \in [t]:\, \chi_r = 1}{\textbf{majority}}\, ans_r(x,i).$$

Analysis: For $x \in \{0,1\}^n$, $i \in [n]$ and $S \subseteq [m]$, let $adv(S,x,i) \stackrel{\text{def}}{=} \sum_{r \in [t]:\chi_r = 1} adv_r(x,i)$. We will choose p, so that for all $x \in \{0,1\}^n$ and $i \in [n]$,

$$\Pr_S[adv(S,x,i) \leq 0] \;\leq\; \frac{1}{3}. \tag{4}$$

Here, the probability is over the sets S produced by our sampling algorithm above. Fix $x \in \{0,1\}^n$ and $i \in [n]$, and write $adv(S)$ instead of $adv(S,x,i)$. Then, $adv(S) = \sum_{r \in [t]} adv_r \chi_r$, and since the decoding algorithm gives the correct answer with probability at least $\frac{1}{2} + \epsilon$, we have $\mathop{\mathbf{E}}_S[adv(S)] = \sum_{r \in [t]} adv_r \,\mathbf{E}[\chi_r] \geq 2\epsilon t \cdot p^q$. Thus, the average advantage is bounded away from 0. But (4) says that the advantage is positive with high probability. For this, we will bound the variance of $adv(S)$:

$$\begin{aligned} \mathbf{var}[adv(S)] &= \mathbf{E}[adv(S)^2] - \mathbf{E}[adv(S)]^2 \\ &= \sum_{r,r' \in [t]} \mathbf{E}[adv_r \chi_r\, adv_{r'} \chi_{r'}] - \mathbf{E}[adv_r \chi_r]\,\mathbf{E}[adv_{r'} \chi_{r'}] \\ &= \sum_{r,r' \in [t]} adv_r\, adv_{r'} (\mathbf{E}[\chi_r \chi_{r'}] - \mathbf{E}[\chi_r]\,\mathbf{E}[\chi_{r'}]). \end{aligned}$$

If $r \neq r'$ and $e_r \cap e_{r'} = \emptyset$, then χ_r and $\chi_{r'}$ are independent, and $\mathbf{E}[\chi_r \chi_{r'}] - \mathbf{E}[\chi_r]\,\mathbf{E}[\chi_{r'}] = 0$. Furthermore, since $\mathbf{E}[\chi_r \chi_{r'}]$ and $\mathbf{E}[\chi_r]\,\mathbf{E}[\chi_{r'}]$ lie in the range $[0, p^q]$, we always have

$$adv_r\, adv_{r'}(\mathbf{E}[\chi_r \chi_{r'}] - \mathbf{E}[\chi_r]\,\mathbf{E}[\chi_{r'}]) \leq p^q.$$

Thus, $\mathbf{var}[adv(S)] \leq p^q \cdot \sum_{r,r' \in [t]: e_r \cap e_{r'} \neq \emptyset} 1 \;\leq\; p^q q \left(\frac{c}{m}\right) t^2,$

where the last inequality follows from (3): $|(r, r') : e_r \cap e_{r'} \neq \emptyset| \leq \sum_{j \in [m]} \deg^2 j \leq (\max_j \deg j) \cdot (\sum_j \deg j) \leq (ct/m) \cdot (qt)$.

Using Chebyshev's inequality, we have

$$\Pr[adv(S) \leq 0] \leq \frac{\mathbf{var}[adv(S)]}{\mathbf{E}[adv(S)]^2} \leq \frac{qc}{4\epsilon^2 p^q m}.$$

Now, let $p \stackrel{\text{def}}{=} \left(\frac{qc}{\epsilon^2 m}\right)^{1/q}$. Since $t \geq 10m$, $\Pr[adv(S) \leq 0] \;\leq\; \frac{1}{4} \;\leq\; \frac{1}{3}$, as required in (4).

Proof of Theorem 1: By Lemma 1.1, we get $pm \log_2 |\Sigma| \;\geq\; n(1 - H(\frac{2}{3}))$, that is

$$m \;\geq\; \left(\frac{\epsilon^2}{qc}\right)^{1/(q-1)} \left[\frac{n(1 - H(\frac{2}{3}))}{\log_2 |\Sigma|}\right]^{q/(q-1)}.$$

3 General Codes

Recall that our overall goal is to show that after picking a small number of bits of the codeword $C(x)$, there is a method for predicting x_i with high probability for any $i \in [n]$. The locations to be read from $C(x)$ are randomly chosen, according to a distribution independent of i. In the previous section, these bits were used only in the sampling procedure for predicting x_i. Now, however, we will use these bits for two purposes: first, we will read (expected) $(q-1)pm$ bits to construct the Δ-smoothener Z; then, we will read (expected) pm bits to predict the bits of x. Thus, we read a total of qpm bits on the average and predict each bit correctly (in the sense of Lemma 1.1) with probability at least $\frac{2}{3}$.

The decoding algorithm: Before we proceed further, we need to look more closely at the decoding algorithm, and introduce some notation. Fix $i \in [n]$. We can model the decoding algorithm d as a collection of decision trees. Each such tree probes the code word $C(x)$ (adaptively) at most q times and returns a guess for x_i. We will make two assumptions about the decoding algorithm.

A1: There are t ($\geq 10m$) decision trees: $\{\Gamma_r\}_{r \in [t]}$. The randomized decoding algorithm picks one of them at random (uniformly, each with probability $\frac{1}{t}$), and guesses x_i to be the answer returned by that tree.

A2: No location of the table is probed twice along any path of a decision tree.

It will be convenient to extend the usual definition of a decision tree a little: some of the nodes of the tree will be designated as *virtual* nodes. When the algorithm reaches a virtual node, it assumes that the entry of the codeword in the location probed by that node is 0, and does not explicitly read any bit of $C(x)$. We then say that algorithm has made a *virtual probe*. When the node reached

is not virtual, the algorithm probes $C(x)$ as usual, and we say that it has made a *real probe*. We require that the total number of nodes (real plus virtual) on any path of the tree be at most q. For a decision tree Γ_r of d, the path now taken in the tree gives rise to a sequence $e_r = \langle a_1, a_2, \ldots, a_k \rangle$, $(k \leq q)$, of indices in $[m]$, some of which might correspond to virtual probes. We use $e_r[j]$ to denote the jth element of e_r. For $a \in [m]$, let

$$\deg_j(a) = |\{r \in [t] : \text{the } j\text{th probe of } e_r \text{ is a real probe and } e_r[j] = a\}|. \tag{5}$$

Let $\deg(a) = \sum_{j=1}^{q} \deg_j(a)$. We will also use an extension of this notation. For $R \subseteq [t]$, let

$$\deg_j(a \mid R) = |\{r \in R : \text{the } j\text{th probe of } e_r \text{ is a real probe and } e_r[j] = a\}|;$$

let $\deg(a \mid R) = \sum_{j=1}^{q} \deg_j(a \mid R)$.

Remark: Note that these definitions depend on the decoding algorithm and the code word being processed. For the rest of this section, fix an input $x \in \{0,1\}^n$ and a decoding algorithm d that guesses x_i by probing the codeword $C(x)$ at most q times.

Definition 3.1 (Δ-smooth decoder) *We say that the decoding algorithm d is Δ-smooth (w.r.t. the codeword $C(x)$) if there is a set $\mathsf{Good} \subseteq [t]$ such that*

$$\begin{aligned} |\mathsf{Good}| &\geq \left(1 - \frac{\epsilon}{2}\right) t, \qquad \textit{and} \\ \forall a \in [m] \forall k \in [q] \ \deg_k(a \mid \mathsf{Good}) &\leq \left(\frac{100q^2}{\epsilon p}\right)^{k-1} \Delta. \end{aligned} \tag{6}$$

Our motivation for considering Δ-smooth decoders comes from the the following *sampling procedure* and the lemma that follows it. Let

$$\Delta \stackrel{\text{def}}{=} \frac{2t}{\delta m} \text{ and } \ p \stackrel{\text{def}}{=} \left(\frac{100}{\epsilon}\right)^{\frac{q+1}{q}} \left(\frac{q^{2q-1}}{\delta m}\right)^{\frac{1}{q}}.$$

Sampling: Let S be the random subset of $[m]$ obtained by picking each $a \in [m]$ with probability p independently.

For the given locally decodable code, fix an adaptive (q, δ, ϵ)-decoder d. From the sample S and this decoder d, we define the prediction algorithm $\mathcal{A}_d$ to be used in Lemma 1.1:

Prediction Algorithm $\mathcal{A}_d$: For $r \in [t]$, let χ_r be the 0-1 random variable (dependent on S) defined by

$$\Pr[\chi_r = 1 \mid S = \hat{S}] = \begin{cases} 0 & \text{if } e_r \not\subseteq \hat{S} \\ p^{q-|e_r|} & \text{if } e_r \subseteq \hat{S} \end{cases}.$$

Let

$$\mathcal{A}_d(i, C(x)[S], S) \stackrel{\text{def}}{=} \underset{r \in [t]: \chi_r = 1}{\textbf{majority}} \ ans_r.$$

Note:

1. By '$e_r \subseteq \hat{S}$' we mean that all *real* probes of e_r are in $\hat{S}$, and by $|e_r|$ we mean the number of *real* probes into $C(x)$ made by the rth decision tree.

2. The additional random choices required in the case '$e_r \subseteq \hat{S}$' are made independently for different r.

3. Our definition implies that $\Pr[\chi_r = 1] = p^q$, for all $r \in [t]$.

Lemma 3.1 *If d is Δ-smooth and d guesses x_i correctly with probability at least $\frac{1}{2} + \epsilon$, then*

$$\Pr_S[\mathcal{A}_d(i, C(x)[S], S) = x_i] \;\geq\; \frac{9}{10}.$$

Proof: Fix $x \in \{0,1\}^n$ and $i \in [n]$. Recall (cf. Definition 2.1) that $ans_r \in \{0,1\}$ denotes the answer returned by d by probing $C(x)$ when its random string is r and that $adv_r = +1$ if $ans_r = x_i$ and $adv_r = -1$ if $ans_r \neq x_i$. Let

$$adv(S) \stackrel{\text{def}}{=} \sum_{r \in [t]} adv_r \cdot \chi_r.$$

Our goal, then, is to show that $\Pr_S[adv(S) \leq 0] \leq \frac{1}{10}$. Let $adv_{\mathsf{G}}(S) \stackrel{\text{def}}{=} \sum_{r \in \mathsf{Good}} adv_r \cdot \chi_r$. Then,

$$adv(S) = adv_{\mathsf{G}}(S) + \sum_{r \notin \mathsf{Good}} adv_r \cdot \chi_r \geq adv_{\mathsf{G}}(S) - \sum_{r \notin \mathsf{Good}} |adv_r| \cdot \chi_r \geq adv_{\mathsf{G}}(S) - \frac{1}{2}\epsilon p^q t.$$

So, it suffices to show that

$$\Pr[adv_{\mathsf{G}}(S) \leq \frac{1}{2}\epsilon p^q t] \;\leq\; \frac{1}{10}. \tag{7}$$

For this, we will use Chebyshev's inequality. We have

$$\begin{aligned}
\mathbf{E}[adv_{\mathsf{G}}(S)] &= \mathbf{E}[adv(S)] - \sum_{r \notin \mathsf{Good}} adv_r \cdot \mathbf{E}[\chi_r] \geq 2\epsilon p^q t - \frac{1}{2}\epsilon p^q t = \frac{3}{2}\epsilon p^q t; \\
\mathbf{var}[adv_{\mathsf{G}}(S)] &= \mathbf{E}[adv_{\mathsf{G}}(S)^2] - \mathbf{E}[adv_{\mathsf{G}}(S)]^2 \\
&= \sum_{r,r' \in \mathsf{Good}} \mathbf{E}[adv_r \chi_r adv_{r'} \chi_{r'}] - \mathbf{E}[adv_r \chi_r]\,\mathbf{E}[adv_{r'} \chi_{r'}] \\
&= \sum_{r,r' \in \mathsf{Good}} adv_r\, adv_{r'} (\mathbf{E}[\chi_r \chi_{r'}] - \mathbf{E}[\chi_r]\,\mathbf{E}[\chi_{r'}]).
\end{aligned}$$

If $r = r'$, then $\mathbf{E}[\chi_r \chi_{r'}] - \mathbf{E}[\chi_r]\,\mathbf{E}[\chi_{r'}] = p^q - p^{2q} \leq p^q$. If $r \neq r'$, and e_r and $e_{r'}$ have no real probe in common, then χ_r and $\chi_{r'}$ are independent, and $\mathbf{E}[\chi_r \chi_{r'}] - \mathbf{E}[\chi_r]\,\mathbf{E}[\chi_{r'}] = 0$. If $r \neq r'$, and e_r and $e_{r'}$ do have a real probe in common, let probe k be the first real probe of e_r that appears as a (real) probe of $e_{r'}$. Then, since $|e_r \cup e_{r'}| \geq q + k - 1$, $|\,\mathbf{E}[\chi_r \chi_{r'}] - \mathbf{E}[\chi_r]\,\mathbf{E}[\chi_{r'}]| \;\leq\; p^{q+k-1} - p^{2q} \;\leq\; p^{q+k-1}$. Note that the case of $k = 1$ includes the case $r = r'$. By considering the terms corresponding to the different k separately, we obtain

$$\begin{aligned}
\mathbf{var}[adv_{\mathsf{G}}(S)] &\leq \sum_{k=1}^{q} \sum_{a \in [m]} \deg_k(a \mid \mathsf{Good}) \deg(a \mid \mathsf{Good}) p^{q+k-1} \\
&\leq \Delta p^q \sum_{k=1}^{q} \left(\frac{100q^2}{\epsilon}\right)^{k-1} \sum_{a \in [m]} \deg(a \mid \mathsf{Good}) \qquad (8) \\
&\leq \Delta p^q \sum_{k=1}^{q} \left(\frac{100q^2}{\epsilon}\right)^{k-1} qt \\
&\leq 2\Delta q p^q \left(\frac{100q^2}{\epsilon}\right)^{q-1} t.
\end{aligned}$$

(To justify (8), we used the assumption that d is Δ-smooth.) By Chebyshev's inequality,

$$\Pr[adv_{\mathsf{G}}(S) \leq \frac{1}{2}\epsilon p^q t] \leq \frac{\mathbf{var}[adv(S)]}{(\mathbf{E}[adv_{\mathsf{G}}(S)] - \frac{1}{2}\epsilon p^q t)^2} \leq \frac{\mathbf{var}[adv(S)]}{(\epsilon p^q t)^2} \leq \left(\frac{2\Delta q}{\epsilon^2 p^q t}\right)\left(\frac{100q^2}{\epsilon}\right)^{q-1}.$$

Recall that $\Delta = \frac{2t}{\delta m}$, $t \geq 10m$ and $p = \left(\frac{100}{\epsilon}\right)^{\frac{q+1}{q}}\left(\frac{q^{2q-1}}{\delta m}\right)^{\frac{1}{q}}$. Thus, $\Pr[adv_{\mathsf{G}}(S) \leq \frac{1}{2}\epsilon p^q t] \leq \frac{1}{10}$, as required in (7). ■

Definition 3.2 (Δ-smoothener) *Let $Z \subseteq [m]$. Then, the smoothening of d by Z, denoted by $d*Z$, is the decoding algorithm obtained from d by designating all nodes where a probe is made to a location in Z as* virtual nodes. *(Nodes that were already virtual in d continue to be virtual in $d*Z$.) We say that $Z \subseteq [m]$ is a Δ-smoothener for d (w.r.t. the codeword $C(x)$) if $d*Z$ is Δ-smooth.*

Lemma .2 *There is a randomized procedure that probes the codeword $C(x)$ at $(q-1)pm$ bits on the average (according to a distribution independent of i) and using contents of those locations and the decoding algorithm d, returns a set Z ($|Z| \leq \delta m$), such that with probability at least $\frac{4}{5}$, $d*Z$ is Δ-smooth.*

Proof of Theorem 2: First pick a sample of bits of $C(x)$ at random as in Lemma .2. Then, given i, construct the Δ-smoothener Z for the decoding algorithm d that predicts x_i by probing $C(x)$ at most q times. Then, pick the sample of S of pm bits for the sampling procedure described at the beginning of this section. Now, let $\mathcal{A}_{d*Z}(i, C(x)[S], S)$ be the guess for x_i. Since $|Z| \leq \delta m$, the query algorithm $d*Z$ guesses x_i correctly with probability at least $\frac{1}{2}+\epsilon$. By Lemma .2 with probability $\frac{4}{5}$, $d*Z$ is Δ-smooth. Then, by Lemma 3.1,

$$\Pr[\mathcal{A}_{d*Z}(i, C(x)[S], S) = x_i \mid d*Z \text{ is } \Delta\text{-smooth}] \geq \frac{9}{10}.$$

Thus, with probability at least $\left(\frac{4}{5}\right)\left(\frac{9}{10}\right) \geq \frac{2}{3}$, this method predicts x_i correctly, after reading qpm bits on the average. By Lemma 1.1, $qpm \log_2 |\Sigma| \geq n(1 - H(\frac{2}{3}))$, that is,

$$m \geq \left(\frac{\epsilon}{100}\right)^{\frac{q+1}{q-1}}\left(\frac{1}{q}\right)^{\frac{3q-1}{q-1}}\delta^{\frac{1}{q-1}}\left[\frac{n(1-H(\frac{2}{3}))}{\log_2|\Sigma|}\right]^{\frac{q}{q-1}}.$$

■

References

[Am] Amabainis, A.: Upper Bounds on the Communication Complexity of Private Information Retrieval, *In Proc. ICALP, pp. 401 – 407, 1997.*

[KT] Katz, J., and Trevisan, L.: On the Efficiency of Local Decoding Procedures for Error-Correcting Codes. *In Proc. 32nd ACM Symposium on Theory of Computing (STOC), 2000.*

[BFLS] Babai, L., Fortnow, L., Levin, L., and Szegedy, M.: Checking Computations in Polylogartithmic Time. *In Proc. 23rd ACM Symposium on Theory of Computing (STOC), pp. 21 – 31, 1991.*

[BFNW] Babai, L., Fortnow, L., Nisan, N., and Wigderson, A.: BPP has Subexponential Time Simulations unless EXP-TIME has publishable proofs. *Computational Complexity, 3(4):307-318, 1993.*

[BI] Beimel, A. and Ishai, Y. : Information-Theoretic Private Information Retrieval: A Unified Construction, *ECCC TR01-015, 2001.*

[CGKS] Chor, B., Goldreich, O., Kushilevitz, E., and Sudan, M.: Private Information Retrieval, *JACM, 45(6), 1998.*

[GLRSW] Gemmel, P., Lipton, R., Rubinfeld, R., Sudan, M., and Wigderson, A.: Self-Testing/Correcting for Polynomials and for Approximate Functions. *In Proc. 23rd STOC, pp. 32 – 42, 1991.*

[GS] Gemmel, P. and Sudan, M. : Highly Resilient Correctors for Polynomials. *Information Processing Letters, 43(4):169–174, 1992.*

[G] Goldreich, O. : Personal Communication cited in [KT], 1999.

[Ma] Mann, E. : Private Access to Distributed Information. *Master's Thesis, Technion, 1998.*

[STV] Sudan, M, Trevisan, L., and Vadhan, S. : Pseudo-random Generators without the XOR Lemma. *in Proc. 31st STOC, pp. 537 – 546, 1999*

A Proof of Lemma 1.1

Pick X uniformly at random from $\{0,1\}^n$ (independent of S). Then,

$$H[X \mid C(x)[S], S] \;\leq\; \sum_{i=1}^{n} H[X_i \mid C(x)[S], S] \;\leq\; nH\left(\frac{2}{3}\right).$$

On the other hand,

$$H[X \mid C(X)[S], S] \;\geq\; H[X \mid S] - H[C(X)[S] \mid S] \;\geq\; n - \mathbf{E}[|S|]\log_2|\Sigma|.$$

B Constructing the Δ-smoothener

B.1 Case $q = 2$

A natural first attempt at obtaining a Δ-smoothener is to identify all locations in $[m]$ that are probed heavily by d. Now, the first probe is non-adaptive, so identifying locations a, with $\deg_1(a) \geq \Delta$ is not a problem. Let $Z_1 \stackrel{\text{def}}{=} \{a : \deg_1(a) \geq \Delta\}$. (Here $\deg_1(a)$ is with respect to d.) We can try to do the same thing for the second probe, but, then, we run into two problems.

1. The second probe depends on what the algorithms reads in its first probe. Hence, for different inputs, different locations can be probed heavily in the second probe.

2. Even if we identify the set of these locations, say Z_2, we cannot guarantee that the algorithm $d * (Z_1 \cup Z_2)$ probes no location heavily. For, setting locations in Z_2 to 0 influences the first probe as well. As a result, the algorithm can now start probing some other locations heavily.

We will now show how one can get around these difficulties. Consider the algorithm $d * Z_1$. We will use sampling to identify the locations that are probed heavily by this algorithm in its second probe. Let T be a random subset obtained by picking each element of $[m]$ independently with probability p. For $r \in [t]$, let χ_r be a random variable (dependent on T), defined by

$$\Pr[\chi_r = 1 \mid T = \hat{T}] = \begin{cases} 1 & \text{if } e_r[1] \text{ is a real probe and } e_r[1] \in T \\ p & \text{if } e_r[1] \text{ is a virtual probe} \\ 0 & \text{otherwise} \end{cases}.$$

Now, let

$$\begin{aligned} \tilde{\deg}_2(a) &\stackrel{\text{def}}{=} |\{r \in [t] : e_r[2] = a \text{ and } \chi_r = 1\}| \\ \text{and } Z_2 &\stackrel{\text{def}}{=} \{a \in [m] : \tilde{\deg}(a) \geq \left(\frac{100}{\epsilon}\right)\Delta\}. \end{aligned}$$

Finally, let $Z \stackrel{\text{def}}{=} Z_1 \cup Z_2$. We will show that Z is a Δ-smoothener for d.

Let us verify that $|Z| \leq \delta m$: $|Z_1| \leq \frac{t}{\Delta} \leq \frac{\delta m}{2}$; similarly, since $\deg_2(a) \geq \left(\frac{100}{\epsilon}\right)\Delta$ for all $a \in Z_2$, and $\sum_a \deg_2(a) \leq t$, we have $|Z_2| \leq \frac{t}{(100/\epsilon)\Delta} \leq \frac{\delta m}{2}$.

Let us, first, deal with the second difficulty mentioned above. How many sequences e_r could have 'changed direction' (that is, $e_r[2]$ is not the same in $d * Z_1$ and $d * (Z_1 \cup Z_2)$) because their first probe fell inside Z_2? If the first probe of e_r falls in Z_1 (or if e_r did not make any probes at all), then designating probes to locations in Z_2 as virtual has no effect on e_r. So, e_r 'changes direction' only if $e_r[1] \in Z_2 - Z_1$. Let

$$\mathsf{Bad}_1 \stackrel{\text{def}}{=} \{r \in [t] : e_r \text{ is different in } d * Z_1 \text{ and } d * (Z_1 \cup Z_2)\ \}.$$

Since $\deg_1(a) \leq \Delta$ for all $a \notin Z_1$, we have, $|\mathsf{Bad}_1| \leq \Delta \cdot |Z_2| \leq \left(\frac{\epsilon}{100}\right) t$. Thus, the number random strings that contribute to the second problem is small, and we can afford to omit them from the set Good (in Definition 3.1).

Now, let us return to the first problem. We have used a randomized method to identify locations that are read heavily in the second probe (by $d * Z_1$). We say that *the method errs* on a if $\deg_2(a) \geq \left(\frac{400}{\epsilon p}\right)\Delta$ (w.r.t. $d * Z_1$), but $a \notin Z_2$, that is, a *is* a heavily probed location, but our sample fails to detect this. Let

$$\mathsf{Bad}_2 \stackrel{\text{def}}{=} \{r : \text{the method errs on } e_r[2]\}.$$

We will show that Bad_2 is small, so we can afford to discard it when we define Good.

Claim 1 *For all a,* $\Pr[\text{the method errs on } a] \leq \frac{\epsilon}{100}$.

Before we prove this claim, let us complete the argument for showing that Z is a Δ-smoothener. Let $\mathsf{Good} \stackrel{\text{def}}{=} [t] - \mathsf{Bad}_1 - \mathsf{Bad}_2$. We need to show that Good is large with high probability. By the above claim,

$$\mathbf{E}[|\mathsf{Bad}_2|] \leq \sum_a \deg_2(a) \Pr[\text{the method errs on } a] \leq \left(\frac{\epsilon}{100}\right) t.$$

By Markov's inequality, $\Pr[|\mathsf{Bad}_2| \geq \frac{49\epsilon}{100}] \leq \frac{1}{49}$. Thus, with probability at least $\frac{48}{49}$, $|\mathsf{Good}| \geq (1 - \frac{\epsilon}{2})t$. To satisfy Definition 3.1, we also need to show that $\deg_2(a \mid \mathsf{Good}) \leq \left(\frac{400}{\epsilon p}\right)\Delta$ for all $a \in [m]$ (w.r.t. the algorithm $d * Z_1 \cup Z_2$). We have already included in Bad_1 all r in for which e_r changes directions.

So, if $\deg_2(a \mid \mathsf{Good}) > \left(\frac{400}{\epsilon p}\right)\Delta$, then it must be that the sampling method errs on a. But then, all r's for which a is the second probe in $d * (Z_1 \cup Z_2)$ are omitted from Good (either via Bad_2 or via Bad_1) and $\deg_2(a \mid \mathsf{Good}) = 0$—a contradiction.

It remains only to prove the claim. Fix a such that $\deg_2(a) \geq \left(\frac{400}{\epsilon p}\right)\Delta$ (w.r.t. $d * Z_1$). Then,

$$\begin{aligned} \mathbf{E}[\tilde{\deg}_2(a)] &> \left(\frac{400}{\epsilon}\right)\Delta \\ \text{and } \mathbf{var}[\tilde{\deg}_2(a)] &= \sum_{r,r'}(\mathbf{E}[\chi_r]\,\mathbf{E}[\chi_{r'}] - \mathbf{E}[\chi_r\chi_{r'}]) \\ &\leq \sum_r E[\chi_r] + \sum_{r\neq r'}(\mathbf{E}[\chi_r\chi_{r'}] - \mathbf{E}[\chi_r]\,\mathbf{E}[\chi_{r'}]). \end{aligned}$$

(In these formulas, r and r' range over random strings for which the algorithm $d * Z_1$ makes its second probe at a.) Now, if the first probe of e_r and the first probe of $e_{r'}$ are different, or if their first probe is virtual, then χ_r and $\chi_{r'}$ are independent and $\mathbf{E}[\chi_r\chi_{r'}] - \mathbf{E}[\chi_r]\,\mathbf{E}[\chi_{r'}] = 0$. Otherwise, $\mathbf{E}[\chi_r\chi_{r'}] - \mathbf{E}[\chi_r]\,\mathbf{E}[\chi_{r'}] \leq \mathbf{E}[\chi_r\chi_{r'}] \leq \mathbf{E}[\chi_r]$. Thus,

$$\mathbf{var}[\tilde{\deg}(a)] \leq \sum_r \mathbf{E}[\chi_r] + \sum_r(\Delta - 1)\,\mathbf{E}[\chi_r] \leq \Delta\,\mathbf{E}[\tilde{\deg}(a)].$$

By Chebyshev's inequality,

$$\begin{aligned} \Pr[\tilde{\deg}(a) < \left(\frac{100}{\epsilon}\right)\Delta] &\leq \frac{\Delta\,\mathbf{E}[\tilde{\deg}(a)]}{(\mathbf{E}[\tilde{\deg}(a)] - (100/\epsilon)\Delta)^2} \\ &\leq \frac{\Delta\,\mathbf{E}[\tilde{\deg}(a)]}{(3/4)^2\,\mathbf{E}[\tilde{\deg}(a)]^2} \\ &\leq \left(\frac{4}{3}\right)^2\frac{\epsilon}{400} \\ &\leq \frac{\epsilon}{100}. \end{aligned}$$

B.2 Case $q > 2$

Fix $i \in [n]$ and let $d \equiv \{\Gamma_r\}_{r\in[t]}$ be the decoding algorithm that predicts x_i by probing $C(x)$ at most q times. Our set Z will be the union of sets Z_k ($k \in [q]$). These sets will be generated by probing the code word $C(x)$ randomly. We wish to include in Z_k all locations probed heavily in the kth probe. As we shall see, we will succeed in this only approximately. To determine Z_k, we use the algorithm $d * (\bigcup_{i=1}^{k-1} Z_i)$.

Vertices of Z_1 can be determined without probing $C(x)$. For each other i, Z_i is determined by choosing a random set T_i of $[m]$ and running the algorithm $d * (\bigcup_{j=1}^{i-1} Z_j)$ on this random set and include in Z_i all locations a that are probed highly in the ith probe. Let us assume that we have determined $Z_1,\ldots,Z_{k-1}$ and we wish to determine Z_k.

We will set aside a small number of random strings r from $[t]$, so that for the remaining random strings the algorithm $d * Z$ does not probe any location heavily. We wanted to include in Z_k all locations $a \in [m]$ such that $\deg_k(a) \geq \left(\frac{100q^2}{\epsilon p}\right)^{k-1}\Delta$ w.r.t. the algorithm $d * \bigcup_{i=1}^{k-1} Z_i$. In our randomized approximation of this ideal Z_k, we cannot guarantee that all such locations are picked. So, to compensate for this, we will have to set aside a set N_k of random strings that cause $d*\bigcup_{k'=1}^{k-1} Z_{k'}$ to make their kth probe into locations that Z_k errs on. The following pseudocode below

describes how the sets N_k are defined in order that $\deg_k(a \mid [t] - \bigcup_{k'=1}^{k} N_{k'})$ (w.r.t. $d * \bigcup_{k'=1}^{k} Z_{k'}$) is small for all a. However, what we really need is that $\deg_k(a \mid \mathsf{Good})$ is small w.r.t. $d * Z$. The problem is that when we add $Z_{k+1}, Z_{k+2}, \ldots, Z_q$ to Z, some additional locations are assumed to contain a 0. This influences even earlier probes of sequences that probe these locations, and can cause these sequences to change the locations where they make their kth probe. So, ensuring that $\deg_k(a \mid [t] - \bigcup_{k'=1}^{k} N_{k'})$ (w.r.t. $d * \bigcup_{k'=1}^{k} Z_{k'}$) is small does not automatically guarantee that $\deg_k(a \mid [t] - \bigcup_{k'=1}^{q} N_{k'})$ (w.r.t. $d * Z$) is small. To get around this, we will have to set aside some more random strings, P_k.

Recall that for $r \in [t]$, e_r is the sequence of locations probed by the algorithm d. We will also need initial segments of these sequences: let e_r^k ($k \in [q]$) be the subsequence of e_r^k consisting of the first k probes.

- Initially, $\mathsf{Good} = [t]$.
- For $k = 1, \ldots, q$
  For $a = 1, \ldots, m$

  At this point, $\{Z_i, N_i, P_i\}_{i=1}^{k-1}$ have already been determined; $\mathsf{Good} = [t] - \bigcup_{i=1}^{k-1}(N_i \cup P_i)$. Consider the algorithm $d * \bigcup_{i=1}^{k-1} Z_i$. With respect to this algorithm, the following invariant holds:

$$\mathbf{I}: \ \forall a \in [m] \forall i < k \ \deg_i(a \mid \mathsf{Good}) \ \leq \ \left(\frac{100q^2}{\epsilon p}\right)^{i-1} \Delta.$$

  Constructing Z_k: Let T_k be the random set obtained by including each $a \in [m]$ in it independently with probability p. For $r \in [t]$, $k \in [q]$, let χ_r^k be the 0-1 random variable (dependent on T_k), defined by

$$\Pr[\chi_r^k = 1 \mid T_k = \hat{T}] \ \stackrel{\text{def}}{=} \ \begin{cases} 0 & \text{if } e_r^{k-1} \not\subseteq \hat{T} \\ p^{k-1-|e_r^{k-1}|} & \text{if } e_r^{k-1} \subseteq \hat{T} \end{cases}.$$

  Here, by '$e_r^{k-1} \subseteq \hat{T}$', we mean that all *real* probes of e_r^{k-1} are in $\hat{T}$; by $|e_r^{k-1}|$, we mean the number of real probes of e_r^{k-1}. Also, the random choices required when $e_r^{k-1} \subseteq \hat{T}$ are made independently for different r. Let

$$\begin{aligned} D_k(a) &\stackrel{\text{def}}{=} \{r \in \mathsf{Good} : e_r[k] = a \wedge a \notin \bigcup_{i=1}^{k-1} Z_i\}; \\ \tilde{\deg}_k(a) &\stackrel{\text{def}}{=} \sum_{r \in D_k(a)} \chi_r^k; \\ Z_k &\stackrel{\text{def}}{=} \{a \in [m] : \tilde{\deg}_k(a) \geq \frac{1}{5}\left(\frac{100q^2}{\epsilon}\right)^{k-1} \Delta\}. \end{aligned}$$

  Constructing N_k, P_k: We say that Z_k errs on $a \in [m]$ if $|D_k(a) \geq \left(\frac{100q^2}{\epsilon p}\right)^{k-1} \Delta$ and $a \notin Z_k$.

$$\begin{aligned} N_k &= \{r \in \mathsf{Good} : Z_k \text{ errs on } e_r[k]\}; \\ P_k &= \{r \in \mathsf{Good} : e_r[k'] \in Z_k, \text{ for some } k' < k\}; \\ \mathsf{Good} &= \mathsf{Good} - N_k - P_k; \end{aligned}$$

Lemma B.1
(a) If $|D_k(a)| \geq \left(\frac{100q^2}{\epsilon p}\right)^{k-1} \Delta$, *then* $\Pr[\tilde{\deg}_k(a) < \frac{1}{5}\left(\frac{100q^2}{\epsilon}\right)^{k-1} \Delta] \leq \frac{\epsilon}{20 \cdot q}$.
(b) If $|D_k(a)| < \frac{1}{10}\left(\frac{100q^2}{\epsilon p}\right)^{k-1} \Delta$, *then* $\Pr[\tilde{\deg}_k(a) > \frac{1}{5}\left(\frac{100q^2}{\epsilon}\right)^{k-1} \Delta] \leq \frac{300qp^{k-1}|D_k(a)|}{\left(\frac{100q^2}{\epsilon}\right)^{k} \Delta}$.

Proof: We will use Chebyshev's inequality.

$$\begin{aligned} \mathbf{E}[\tilde{\deg}_k(a)] &= p^{k-1}|D_k(a)|; \\ \mathbf{var}[\tilde{\deg}_k(a)] &= \sum_{\alpha,\beta \in N} \mathbf{E}[\chi_\alpha^k \chi_\beta^k] - \mathbf{E}[\chi_\alpha^k]\,\mathbf{E}[\chi_\beta^k]. \end{aligned} \tag{9}$$

If $\alpha = \beta$, then $\mathbf{E}[\chi_\alpha^k \chi_\beta^k] - \mathbf{E}[\chi_\alpha^k]\,\mathbf{E}[\chi_\beta^k] \le \mathbf{E}[\chi_\alpha^k] = p^{k-1}$. If $\alpha \neq \beta$ and e_α^{k-1} and e_β^{k-1} have no real probe in common, then χ_α^k and χ_β^k are independent, and $E[\chi_\alpha^k \chi_\beta^k] - E[\chi_\alpha^k]E[\chi_\beta^k] = 0$. If $\alpha \neq \beta$, and e_α^{k-1} and e_β^{k-1} do have a real probe in common, then let probe k' be the first real probe of e_α^{k-1} that appears as a real probe of e_β^{k-1}. Then,

$$\mathbf{E}[\chi_\alpha^k \chi_\beta^k] - \mathbf{E}[\chi_\alpha^k]\,\mathbf{E}[\chi_\beta^k] \;\le\; p^{(k-1)+(k'-1)}.$$

We know from the invariant **I** of the algorithm that

$$\forall j \in [m]\, \forall k' \le k-1 \quad \deg_{k'}(j \mid D_k(a)) \le \left(\frac{100q^2}{\epsilon p}\right)^{k'-1} \Delta.$$

Using this inequality and by considering the terms corresponding to $\alpha = \beta$ and $\alpha \neq \beta$ separately, we have

$$\begin{aligned}
\mathbf{var}[\tilde{\deg}_k(a)] &= \sum_{\alpha \in D_k(a)} p^{k-1} + \sum_{k'=1}^{k-1} \sum_{a \in [m]} p^{(k-1)+(k'-1)} \deg_{k'}(a \mid D_k(a)) \deg(a \mid D_k(a)) \\
&\le \mathbf{E}[\tilde{\deg}_k(a)] + \sum_{k'=1}^{k-1} p^{(k-1)+(k'-1)} \left(\frac{100q^2}{\epsilon p}\right)^{k'-1} \Delta \sum_{a \in [m]} \deg(a \mid D_k(a)) \\
&\le \mathbf{E}[\tilde{\deg}_k(a)] + \sum_{k'=1}^{k-1} \left(\frac{100q^2}{\epsilon}\right)^{k'-1} \Delta \sum_{a \in [m]} p^{k-1} \deg(a \mid D_k(a)) \\
&= \mathbf{E}[\tilde{\deg}_k(a)] + 2\left(\frac{100q^2}{\epsilon}\right)^{k-2} \Delta q\, \mathbf{E}[\tilde{\deg}_k(a)] \\
&\le 3q \left(\frac{100q^2}{\epsilon}\right)^{k-2} \Delta\, \mathbf{E}[\tilde{\deg}_k(a)].
\end{aligned}$$

To prove (a), by Chebyshev's inequality,

$$\begin{aligned}
\Pr[\tilde{\deg}_k(a) < \frac{1}{5}\left(\frac{100q^2}{\epsilon}\right)^{k-1} \Delta] &< \frac{\mathbf{var}[\tilde{\deg}_k(a)]}{(\mathbf{E}[\tilde{\deg}_k(a)] - \frac{1}{5}\left(\frac{100q^2}{\epsilon}\right)^{k-1}\Delta)^2} \\
&\le \frac{\mathbf{var}[\tilde{\deg}_k(a)]}{\left(\frac{4}{5}\right)^2 \mathbf{E}[\tilde{\deg}_k(a)]^2} \\
&\le \frac{3q\left(\frac{100q^2}{\epsilon}\right)^{k-2} \Delta\, \mathbf{E}[\tilde{\deg}_k(a)]}{\left(\frac{4}{5}\right)^2 \mathbf{E}[\tilde{\deg}_k(a)]^2} \\
&\le \frac{\epsilon}{20 \cdot q}.
\end{aligned}$$

To justify the second and the last inequalities, we used (9).
To prove (b), using Chebyshev's inequality again,

$$\Pr[\tilde{\deg}_k(a) > \frac{1}{5}\left(\frac{100q^2}{\epsilon}\right)^{k-1} \Delta] \;\le\; \frac{3q\left(\frac{100q^2}{\epsilon}\right)^{k-2} \Delta\, \mathbf{E}[\tilde{\deg}_k(a)]}{\left(\frac{1}{10}\left(\frac{100q^2}{\epsilon}\right)^{k-1}\Delta\right)^2}$$

$$\leq \frac{300qp^{k-1}|D_k(a)|}{\left(\frac{100q^2}{\epsilon}\right)^k \Delta}$$

■

Lemma B.2 $\mathbf{E}[|Z_k|] \leq 20\left(\frac{\epsilon p}{100q^2}\right)^{k-1}\frac{t}{\Delta}$.

Proof: Define the subsets H_k and L_k of $[m]$ as follows:

$$H_k \stackrel{\text{def}}{=} \{a \in [m] : D_k(a) \geq \frac{1}{10}\left(\frac{100q^2}{\epsilon p}\right)^{k-1}\Delta\}.$$

$$L_k \stackrel{\text{def}}{=} \{a \in [m] : D_k(a) < \frac{1}{10}\left(\frac{100q^2}{\epsilon p}\right)^{k-1}\Delta\}.$$

It is easily seen that

$$\begin{aligned} |H_k| &\leq \frac{t}{\frac{1}{10}\left(\frac{100q^2}{\epsilon p}\right)^{k-1}\Delta} \\ &= 10\left(\frac{\epsilon p}{100q^2}\right)^{k-1}\frac{t}{\Delta}. \end{aligned}$$

We know from part (b) of Lemma B.1 that for $a \in L_k$,

$$Pr[a \in Z_k] \leq \frac{300qp^{k-1}|D_k(a)|}{\left(\frac{100q^2}{\epsilon}\right)^k \Delta}$$

Hence,

$$\begin{aligned} \mathbf{E}[|L_k \cap Z_k|] \leq \sum_{a \in L_k} \Pr[a \in Z_k] &\leq \frac{300qp^{k-1}t}{\left(\frac{100q^2}{\epsilon}\right)^k \Delta} \\ &\leq \left(\frac{3\epsilon}{q}\right)\left(\frac{\epsilon p}{100q^2}\right)^{k-1}\frac{t}{\Delta}. \end{aligned}$$

Thus,

$$\begin{aligned} \mathbf{E}[|Z_k|] &\leq |H_k| + E[|L_k \cap Z_k|] \\ &\leq 10\left(\frac{\epsilon p}{100q^2}\right)^{k-1}\frac{t}{\Delta} + \left(\frac{3\epsilon}{q}\right)\left(\frac{\epsilon p}{100q^2}\right)^{k-1}\frac{t}{\Delta} \\ &\leq 20\left(\frac{\epsilon p}{100q^2}\right)^{k-1}\frac{t}{\Delta} \end{aligned}$$

■

Lemma B.3 *For the set $Z \subseteq [m]$ produced by the randomized procedure above:*

(a) $|Z| \leq \delta m$*; and*

(b) Z *is a* Δ*-smoothener for* d *with probability* $\geq \frac{4}{5}$.

Proof:

(a) If $j \in Z_k$, then $\tilde{\deg}_k(j) \geq \frac{1}{5}\left(\frac{100q^2}{\epsilon}\right)^{k-1}$. So we have

$$|Z_k| \;\leq\; \frac{t}{\frac{1}{5}\left(\frac{100q^2}{\epsilon}\right)^{k-1}\Delta} \;\leq\; \frac{t}{2^{k-1}\Delta} \;\leq\; \frac{\delta m}{2^k}$$

since $\Delta = \frac{2t}{\delta m}$. Thus, $|Z| \leq \sum_{i=1}^{q} |Z_i| \leq \delta m$.

(b) It is now straightforward to verify that for all $a \in [m]$, $\deg_k(a \mid \mathsf{Good}) \leq \left(\frac{100q^2}{\epsilon p}\right)^{k-1}\Delta$ w.r.t. $d * Z$. Let $\mathsf{Bad} = [t] - \mathsf{Good}$. It remains to show that $|\mathsf{Bad}|$ is small with high probability. Let $N_k(a) = \{r \in N_k : e_r[k] = a \text{ (w.r.t. } d * \bigcup_{k'=1}^{k-1} Z_{k'})\}$. By Lemma B.1 (applied with $N = N_k(j)$), $\Pr[Z_k \text{ errs on } a] \leq \frac{\epsilon}{20q}$. Thus, $\mathbf{E}[|N_k|] \leq \left(\frac{\epsilon}{20q}\right)t$ and $\mathbf{E}[|\bigcup_{k=1}^{q} N_k|] \leq \left(\frac{\epsilon}{20}\right)t$. Next, we bound P_k. The vertices of Z_k do not belong to $Z_{k'}$ $(k' < k)$, so for $a \in Z_k$, when Z_k is constructed

$$|\{r \in E : e_r[k'] = a\}| \leq \left(\frac{100q^2}{\epsilon p}\right)^{k'-1}\Delta.$$

Thus,

$$\begin{aligned}
\mathbf{E}[|P_k|] &\leq \sum_{k'=1}^{k-1}\left(\frac{100q^2}{\epsilon p}\right)^{k'-1}\Delta\,\mathbf{E}[|Z_k|] \\
&\leq \sum_{k'=1}^{k-1}\left(\frac{100q^2}{\epsilon p}\right)^{k'-1}\Delta\cdot 20\left(\frac{\epsilon p}{100q^2}\right)^{k-1}\frac{t}{\Delta} \\
&\leq 2\left(\frac{100q^2}{\epsilon p}\right)^{k-2}\Delta\cdot 20\left(\frac{\epsilon p}{100q^2}\right)^{k-1}\frac{t}{\Delta} \\
&= \frac{40\epsilon p t}{100q^2} \\
&= \left(\frac{8p}{q}\right)\left(\frac{\epsilon}{20q}\right)t \\
&\leq \left(\frac{\epsilon}{20q}\right)t. \quad \text{(assuming } p < 1/4)
\end{aligned}$$

Hence, $\mathbf{E}[|\bigcup_{k=1}^{q} P_k|] \leq \left(\frac{\epsilon}{20}\right)t$. Thus, $\mathbf{E}[|\mathsf{Bad}|] \leq \left(\frac{\epsilon}{10}\right)t$, and by Markov's inequality $\Pr[|\mathsf{Bad}| \geq \left(\frac{\epsilon}{2}\right)t] \leq \frac{1}{5}$.

■