Electronic Colloquium on Computational Complexity, Revision 2 of Report No. 134 (2010)

A Note on Amplifying the Error-Tolerance of Locally Decodable Codes

Avraham Ben-Aroya * Klim Efremenko† Amnon Ta-Shma‡

Abstract

Trevisan [Tre03] suggested a transformation that allows amplifying the error rate a code can handle. We observe that this transformation, that was suggested in the non-local setting, works also in the local setting and thus gives a generic, simple way to amplify the error-tolerance of locally decodable codes. Specifically, this shows how to transform a locally decodable code that can tolerate a constant fraction of errors to a locally decodable code that can recover from a much higher error-rate, and how to transform such locally decodable codes to *locally list-decodable codes*.

The transformation of [Tre03] involves a simple composition with an *approximately* locally (list) decodable code. Using a construction of such codes by Impagliazzo et al. [IJKW10], the transformation incurs only a negligible growth in the length of the code and in the query complexity.

1 Introduction

Locally decodable codes (LDCs) are codes that allow retrieving any symbol of a message by reading only a constant number of symbols from its codeword, even if a large fraction of the codeword is corrupted. Formally, a code $\mathcal{C}$ is said to be locally decodable with parameters (q, ϵ, δ) if it is possible to recover any symbol x_i of a message x by making at most q queries to $\mathcal{C}(x)$, such that even if a δ fraction of $\mathcal{C}(x)$ is corrupted, the decoding algorithm returns the correct answer with probability at least $1 - \epsilon$.

Locally decodable codes play an important role in many areas in theoretical computer science. While the first formal definition of locally decodable codes was given by Katz and Trevisan [KT00], these codes implicitly appeared in previous works.

The main line of research regarding LDCs seeks to identify the shortest possible code length, in terms of the message length n, while keeping the query complexity, the error-rate and the success probability constant. The Hadamard code is the best-known 2-query locally decodable code and its length is 2^n. For 2-query LDCs tight lower bounds on the

*The Blavatnik School of Computer Science, Tel-Aviv University, Israel, 69978. Supported by the Adams Fellowship Program of the Israel Academy of Sciences and Humanities, by the Israel Science Foundation, by the Wolfson Family Charitable Trust, and by a European Research Council (ERC) Starting Grant.

†The Blavatnik School of Computer Science, Tel-Aviv University, Israel, 69978. Supported by the Israel Science Foundation, by the Wolfson Family Charitable Trust, and by Oded Regev's European Research Council (ERC) Starting Grant.

‡The Blavatnik School of Computer Science, Tel-Aviv University, Israel, 69978. Supported by Israel Science Foundation grant 217/05.

ISSN 1433-8092

code length of $2^{\theta(n)}$ were given in [GKST02] for linear codes and in [KdW03] for general codes. For an arbitrary constant number of queries q, there are weak polynomial bounds, see [KT00, KdW03, Woo07]. The first subexponential LDCs (with a constant number of queries) were obtained by Yekhanin [Yek08]. Yekhanin obtained 3-query LDCs with subexponential length under a highly believable number theoretic conjecture. Later, Efremenko [Efr09], building on Yekhanin [Yek08] and Raghavendra [Rag07], gave an unconditional construction of subexponential length LDCs. This construction also allowed a tradeoff between the number of queries and the codeword length. Subsequently, the query complexity of this construction was improved in [IS08, MFL$^+$10].

The decoding algorithm in all of the aforementioned constructions is *smooth*, i.e., each of its queries is uniformly distributed. The analysis of the decoding algorithm relied on all of the queried symbols being uncorrupted. Using the union bound, one could obtain a decoder with success probability greater than half, only if the error-rate was below $\frac{1}{2q}$.

Another line of research focused on improving the error-tolerance of LDCs. Woodruff [Woo08] showed how to increase the handled error-rate from $\frac{1}{2q}$ to $\frac{1}{q}$ over binary alphabets. Dvir, Gopalan and Yekhanin [DGY10], showed how to handle $\frac{1}{4}$ fraction of errors for the codes of [Efr09]. Ben-Aroya et. al. [BET10] showed the same codes could recover from any error-rate below $\frac{1}{2}$. Gal and Mills [GM09] obtained exponential lower bounds for 3-query LDCs that can tolerate a high error-rate.

When the error-rate is above half of the code's distance, the information in a corrupted codeword is insufficient to uniquely identify the original (uncorrupted) codeword. Thus, in this case, we have to consider *list-decoding*. A code $\mathcal{C}$ is said to be $(1-\alpha, L)$-list-decodable if for every word, the number of codewords within relative distance $1-\alpha$ from that word is at most L. The notion of list-decoding dates back to works by Elias [Eli57] and Wozencraft [Woz58] in the 50s. Roughly speaking, a code $\mathcal{C}$ is *locally* list-decodable if it is $(1-\alpha, L)$-list-decodable, and given a corrupted word w, an index $k \in [L]$ and a target bit j, the decoder returns the j'th message bit of the k'th codeword that is close to w. As expected, there are some subtleties in the definition. The main issue is guaranteeing that for a fixed k, all answers for inputs (k, j) correspond to the same codeword. More formally, a local list-decoding algorithm generates L machines $\{M_k\}$, such that the machine M_k locally decodes one codeword that is close to w, and the machines $\{M_k\}$ together cover all the codewords that are close to w (for a formal definition, see Section 2).

The notion of local list-decoding is central in theoretical computer science. It first implicitly appeared in the celebrated Goldreich-Levin result [GL89], that can be interpreted as a local list-decoding algorithm for the Hadamard code. Later on, many local list-decoding algorithms were studied, especially for Reed-Muller codes [GRS00, AS03, STV01, GKZ08], direct product and XOR codes [IW97, IJK06, IJKW08] and low-rate random codes [KS07, KS09]. In [BET10] it was shown how to locally list-decode the subexponential-length codes of [Efr09] with only a constant number of queries.

Our Result. In this note we show a *generic*, *simple* transformation that takes a locally decodable code $\mathcal{C}$ that can tolerate a low error-rate, and results in a code $\mathcal{C}'$ that can tolerate a much higher fraction of errors. The construction also works in the list-decoding regime, i.e., it can transform any LDC $\mathcal{C}$ to a code $\mathcal{C}'$ which is locally list-decodable from an error-rate of $1-\gamma$, for any $\gamma > 0$. Furthermore, the list-decoder for the new code outputs only a constant

number of codewords.

The transformation was suggested previously by Trevisan [Tre03], who used it to construct list-decodable codes. We observe that this transformation, when used with a locally-decodable code, results in a locally list-decodable code. While the observation is trivial, it appears to have been unnoticed previously.

The transformation is based on the following idea. An error correcting code with relative distance α is a function $\mathcal{C} : \Sigma^n \to \Sigma^{\bar{n}}$ that maps any two different strings, to two strings that differ in at least an α fraction of the coordinates. The decoding algorithm can therefore map any string $\tilde{c}$ with more than $(1 - \alpha/2)n$ agreement with a codeword $c = \mathcal{C}(\lambda)$, to the correct message λ. We can view this as an $\alpha/2$ to 0 error reduction: given a codeword with some $\alpha/2$ fraction of errors, one can correctly recover the original message.[1].

Similarly, one can define a related notion of codes that only amplify the error-tolerance, without completely correcting the corrupted word. That is, one can design a code $\mathcal{C} : \Sigma^n \to \Sigma^{\bar{n}}$, such that given access to a corrupted word $\tilde{c}$ with γn agreement with some codeword $c = \mathcal{C}(\lambda)$, one can compute a message $\tilde{\lambda}$ with some larger $\beta > \gamma$ agreement with λ. We call such a code an *approximately* locally decodable code. When γ is small, several codewords can be γ-close to $\tilde{c}$ and one has to resort to list-decoding. In this case, the code is called an *approximately* locally list-decodable code. Such codes naturally arise in hardness amplification (see, e.g., [IJKW10]). For a formal definition see Section 2.

Now, let us return to the problem of finding a good locally list-decodable code. Our approach is to compose a locally decodable code (handling the $\alpha/2$ to 0 error reduction) with an approximately locally list-decodable code (handling the $\frac{1}{2} - \epsilon$ to $\alpha/2$ error reduction, for binary codes). Namely, we first encode a message λ with a locally-decodable code $\mathcal{C}$ and then encode the result with an approximately locally list-decodable code to get the code $\mathcal{C}'$. To see that it works, assume we are given a word with $\frac{1}{2} + \epsilon$ agreement with some codeword of $\mathcal{C}'$. We first apply the approximate local list-decoder and get a list of words, each with $1 - \alpha/2$ agreement with some codeword of $\mathcal{C}$. We then (uniquely) locally decode each of these corrupted codewords to get λ_i, the i'th symbol of the message λ.

In fact, the local list-decoders of Reed-Muller codes [GRS00, AS03, STV01, GKZ08] and the Hadamard code [GL89], also have this two-step structure, combining an error-reduction step (that does not completely correct the corrupted word) with another unique decoding step. The main difference is that Reed-Muller and Hadamard codes are *locally correctable*, i.e., the first error-reduction step returns a close codeword, instead of close message. Therefore, these two steps can be done implicitly without the use of any general approximate list-decoding mechanism. In our case we present a generic transformation that may work with LDCs that are not known to be locally correctable (e.g., the code of [Efr09]) and we therefore need to compose the code with an approximately locally list-decodable code.

Composing the locally decodable binary codes of [Efr09, MFL$^+$10] with the binary approximately locally list-decodable codes of [IJKW10] we get:

Theorem 1. *For every $r \geq 2$ there exists a binary code of length*

$$\exp(\exp(O(\sqrt[r]{\log n (\log\log n)^{r-1}}))),$$

[1]For a treatment of the related notion of worst-case to average-case reduction and its relationship to error correcting codes see, e.g., [IJKW10].

which is locally list-decodable from an error-rate of $1/2 - \alpha$*. The list-decoding algorithm outputs a list of size* $O(\frac{1}{\alpha^2})$ *and uses at most* $O(\frac{r+\log(1/\epsilon)}{\alpha^3} \cdot 2^r)$ *queries.*

A locally list-decodable code of similar length was given in [BET10]. However, the list size in the list-decoding algorithm of [BET10] was $\mathrm{poly}(n)$, while in Theorem 1 it is constant. The query complexity of the list-decoding algorithm of [BET10] was also worse than that of Theorem 1. On the other hand, the result in [BET10] shows the code of [Efr09] is locally list-decodable, while Theorem 1 only shows that some other (related) code is locally list-decodable, and does not state anything about the original code of [Efr09].

2 Definitions

The *agreement* between strings x and y is the fraction of coordinates i in which $x_i = y_i$. The agreement between x and y is denoted by $\mathrm{Ag}(x, y)$.

Definition 1. *A probabilistic oracle machine* M^w locally outputs a string s with confidence $1 - \epsilon$*, if*

$$\forall i \ \Pr[M^w(i) = s_i] \geq 1 - \epsilon,$$

where the probability is over the randomness of M*.*

Definition 2. *A deterministic oracle machine* M^w locally ϵ-approximates a string s *, if*

$$\Pr_i[M^w(i) = s_i] \geq 1 - \epsilon,$$

where the probability is over a uniformly chosen i*.*

Note that if M^w locally outputs a string s with confidence $1 - \epsilon$ then there is a way to fix its randomness such that it will locally ϵ-approximates a string s. Essentially, M^w locally approximates a string s if it outputs a string that is close to s.

Definition 3 (Local unique decoding)**.** *A code* $\mathcal{C} : \Sigma^n \mapsto \Sigma^{\bar{n}}$ *is* (q, ϵ, δ) locally decodable *if there exists a probabilistic oracle machine* M^w *(the decoding algorithm) with oracle access to a received codeword* w *such that:*

1. *For every message* $\lambda = (\lambda_1, \lambda_2, \ldots \lambda_n) \in \Sigma^n$ *and for every* $w \in \Sigma^{\bar{n}}$ *such that* $\mathrm{Ag}(\mathcal{C}(\lambda), w) \geq 1 - \delta$ *it holds that* M^w *locally outputs* λ *with confidence* $1 - \epsilon$*.*
2. $M^w(i)$ *makes at most* q *queries to* w *for all* $i \in [n]$*.*

It is possible to consider a more relaxed notion of local decoding, where the machine M^w is not required to successfully decode every i. Instead, it is required to succeed on average over i:

Definition 4 (Approximate local unique decoding)**.** *A code* $\mathcal{C}$ *over a field* Σ*,* $\mathcal{C} : \Sigma^n \mapsto \Sigma^{\bar{n}}$ *is* (q, ϵ, δ) approximately locally decodable *if there exists a deterministic oracle machine* M^w *(the decoding algorithm) with oracle access to a received codeword* w *such that:*

1. *For every message* $\lambda = (\lambda_1, \lambda_2, \ldots \lambda_n) \in \Sigma^n$ *and for every* $w \in \Sigma^{\bar{n}}$ *such that* $\mathrm{Ag}(\mathcal{C}(\lambda), w) \geq 1 - \delta$*, it holds that* M^w *locally* ϵ*-approximates* λ*.*

2. *$M^w(i)$ makes at most q queries to w for all $i \in [n]$.*

Although the definitions of locally decodable codes and approximately locally decodable codes are similar, it appears that it is much harder to construct locally decodable codes than approximately locally decodable codes. While there exist constant-query approximately locally decodable codes of polynomial length, no such locally decodable codes are known. Approximately locally decodable are interesting when $\epsilon < \delta$, since the identity code is a $(1, \epsilon, \epsilon)$ approximately locally decodable code.

A code $\mathcal{C}$ is list-decodable if for every word, there are a few codewords near it. Let $\mathcal{C}(y_1), \mathcal{C}(y_2), \ldots, \mathcal{C}(y_L)$ be the list of codewords near a word w. Roughly speaking, a code $\mathcal{C}$ is locally list-decodable if there exists a machine M, that given i, j and an oracle access to the received word w, outputs the jth symbol of y_i. The locality property requires that the machine M makes a few queries to w. Formally:

Definition 5 (Local list-decoding)**.** *Let $\mathcal{C} : \Sigma^n \to \Sigma^{\bar{n}}$ be a code. A set of probabilistic oracle machines $M_1 \ldots M_L$ with oracle queries to w, (α, L, q, ϵ)* locally list-decodes *$\mathcal{C}$ at the word $w \in \Sigma^{\bar{n}}$, if,*

- *Every oracle machine M_j makes at most q queries to the input word w.*
- *For every codeword $c \in \mathcal{C}$ with $\mathrm{Ag}(c, w) \geq \alpha$, there exists some $k \in [L]$, such that M_k^w locally outputs c with confidence $1 - \epsilon$.*

We can also define *approximate* local list-decoding by relaxing the requirement that M_k^w successfully decodes c on every i. Instead, we require successful decoding on average over i.

Definition 6 (Approximate local list-decoding)**.** *Let $\mathcal{C} : \Sigma^n \to \Sigma^{\bar{n}}$ be a code. A set of deterministic oracle machines $M_1 \ldots M_L$ with oracle queries to w, (α, L, q, ϵ)* approximately locally list-decodes *$\mathcal{C}$ at the word $w \in \Sigma^{\bar{n}}$, if,*

- *Every oracle machine M_j makes at most q queries to the input word w.*
- *For every codeword $c \in \mathcal{C}$ with $\mathrm{Ag}(c, w) \geq \alpha$, there exists some $k \in [L]$, such that M_k^w ϵ-approximates c.*

Definition 7 ((Approximately) Locally list-decodable codes with deterministic reconstruction)**.** *Let $\mathcal{C} : \Sigma^n \to \Sigma^{\bar{n}}$ be (α, L) list-decodable. A deterministic algorithm A (α, L, q, ϵ)* (approximately) locally list-decodes *$\mathcal{C}$, if on input n, A outputs oracle machines $M_1 \ldots M_L$ which (α, L, q, ϵ) (approximately) locally list-decode $\mathcal{C}$ at* every *word $w \in \Sigma^{\bar{n}}$.*

The code $\mathcal{C}$ is (α, L) list-decodable and therefore every $w \in \Sigma^{\bar{n}}$ has at most L codewords $c_1, \ldots, c_L$ that are α-close to it. Each such codeword $c_i = \mathcal{C}(\lambda^i)$ is represented by a probabilistic machine M_i such that:

- If the code is locally list-decodable then $\forall j \; M_i(j) = \lambda_j^i$ with probability at least $1 - \epsilon$.
- If the code is approximately locally list-decodable then $M_i(j) = \lambda_j^i$ for at least a $1 - \epsilon$ fraction of the indices j.

The algorithm A outputs L machines that are good for every $w \in \Sigma^{\bar{n}}$. One way to think about it is that $i \in [L]$ is an advice that specifies which of the L solutions corresponds to the codeword we are interested in.

Definition 8 (Locally list-decodable codes with probabilistic reconstruction)**.** *Let $\mathcal{C} : \Sigma^n \to \Sigma^{\bar{n}}$ be (α, L) list-decodable. A probabilistic algorithm A (α, L, q, ϵ)* locally list-decodes *$\mathcal{C}$, if on input n, A outputs probabilistic oracle machines $M_1 \dots M_L$ such that for every word $w \in \Sigma^{\bar{n}}$, with probability $2/3$ over the random coins of A, the machines $M_1 \dots M_L$ locally list-decode $\mathcal{C}$ at w, i.e.,*

$$\forall w \in F^{\bar{n}} \Pr_A \left[\forall \lambda \left(\mathrm{Ag}(\mathcal{C}(\lambda), w) \geq \alpha \Rightarrow \exists i \, \forall j \Pr[M_i(j) = \lambda_j] \geq 1 - \epsilon \right) \right] \geq 2/3.$$

Definition 9 (Approximately locally list-decodable codes with probabilistic reconstruction)**.** *Let $\mathcal{C} : \Sigma^n \to \Sigma^{\bar{n}}$ be (α, L) list-decodable. A probabilistic algorithm A (α, L, q, ϵ)* approximately locally list-decodes *$\mathcal{C}$, if on input n, A outputs deterministic oracle machines $M_1 \dots M_L$ such that for every word $w \in \Sigma^{\bar{n}}$, with probability $2/3$ over the random coins of A, the machines $M_1 \dots M_L$ approximately locally list-decode $\mathcal{C}$ at w, i.e.,*

$$\forall w \in F^{\bar{n}} \Pr_A \left[\forall \lambda \left(\mathrm{Ag}(\mathcal{C}(\lambda), w) \geq \alpha \Rightarrow \exists i \Pr_j [M_i(j) = \lambda_j] \geq 1 - \epsilon \right) \right] \geq 2/3.$$

The best approximately list-decodable codes currently known (to the best of our knowledge) are due to Impagliazzo et al. [IJKW10]. In this note we focus on binary codes, although by using the non-binary codes of [IJKW10] one can also get non-binary list-decodable codes.

Theorem 2 ([IJKW10][2])**.** *For every $\alpha, \epsilon > 0$ there exists a number $f(\alpha, \epsilon)$ such that there exists a code $\mathcal{C}_{\mathrm{App}} : \{0,1\}^n \mapsto \{0,1\}^{f(\alpha,\epsilon) n^5}$ which is $(1/2 + \alpha, O(\frac{1}{\alpha^2}), O(\frac{\log(1/\epsilon)}{\alpha^3}), \epsilon)$ approximately locally list-decodable.*

3 Composition Theorem

Our main observation in this note is that if a code $\mathcal{C}_{\mathrm{LDC}}$ is locally decodable and a code $\mathcal{C}_{\mathrm{App}}$ is approximately locally decodable then by composing these two codes we get a code which is locally decodable, and can tolerate a higher error-rate.

Theorem 3. *Let $\mathcal{C}_{\mathrm{LDC}} : \Sigma_1^n \mapsto \Sigma_2^{N'}$ be (q, ϵ, δ) locally decodable code and let $\mathcal{C}_{\mathrm{App}} : \Sigma_2^{N'} \mapsto \Sigma_3^N$ be an (q', δ, δ') approximately locally decodable code. Then the code $\mathcal{C} = \mathcal{C}_{\mathrm{App}} \circ \mathcal{C}_{\mathrm{LDC}} : \Sigma_1^n \mapsto \Sigma_3^N$ defined by $\mathcal{C}(\lambda) = \mathcal{C}_{\mathrm{App}}(\mathcal{C}_{\mathrm{LDC}}(\lambda))$ is $(q \cdot q', \epsilon, \delta')$ locally decodable.*

Thus, if we have a locally decodable code which can tolerate a small fraction of errors, the above theorem allows us to amplify the error-rate by using an approximately locally decodable code. We have similar theorem for the list-decoding regime:

Theorem 4. *Let $\mathcal{C}_{\mathrm{LDC}} : \Sigma_1^n \mapsto \Sigma_2^{N'}$ be (q, ϵ, δ) locally decodable code and let $\mathcal{C}_{\mathrm{App}} : \Sigma_2^{N'} \mapsto \Sigma_3^N$ be an (α, L, q', δ) approximately locally list-decodable code. Then the code $\mathcal{C} = \mathcal{C}_{\mathrm{App}} \circ \mathcal{C}_{\mathrm{LDC}} : \Sigma_1^n \mapsto \Sigma_3^N$ defined by $\mathcal{C}(\lambda) = \mathcal{C}_{\mathrm{App}}(\mathcal{C}_{\mathrm{LDC}}(\lambda))$ is $(\alpha, q \cdot q', L, \epsilon)$ locally list-decodable.*

[2]The code we use is not explicit in [IJKW10], but it can be deduced from Section 5 in that paper. In Section 5 it is shown that a longer code (the direct-product code, concatenated with Hadamard) is approximately locally list-decodable. However, the same proof carries over when using the derandomized direct-product code (concatenated with Hadamard). The parameter d (of [IJKW10]) is set to 5 (this is affects the exponent in the codeword length). The number of queries is $O(\frac{\log(1/\epsilon)}{\alpha^3})$ since we need to run the Goldreich-Levin algorithm $O(\frac{\log 1/\epsilon}{\alpha})$ times, and each run requires $1/\alpha^2$ queries.

Proof: Let A denote the reconstruction algorithm for the code $\mathcal{C}_{\mathrm{App}}$ and let $D^w : [n] \mapsto \Sigma_1$ denote the unique decoding algorithm for the code $\mathcal{C}_{\mathrm{LDC}}$. The reconstruction algorithm for the code $\mathcal{C}$ works as follows: it first applies the algorithm A to obtain a list of machines $M_1, \ldots, M_L$. For each machine M_j, it outputs the machine Z_j defined by $Z_j^w(i) = D^{M_j^w}(i)$.

The bounds on the number of queries and the list size are immediate. Fix a word $w \in \Sigma_3^N$. The inner reconstruction algorithm A fails with probability at most $\frac{1}{3}$. When it does not fail, we will show that for every codeword with at least α agreement with w, its message is outputted with confidence $1-\epsilon$ by one of the output machines. Suppose that the agreement between $\mathcal{C}_{\mathrm{App}}(\mathcal{C}_{\mathrm{LDC}}(\lambda))$ and w is at least α. Denote $\zeta = \mathcal{C}_{\mathrm{LDC}}(\lambda)$. Since A did not fail, one of the machines M_j^w δ-approximates ζ. Thus, $Z_j^w = D^{M_j^w}$ locally outputs λ with confidence $1-\epsilon$. ■

The above theorem give locally list-decodable codes which improve upon previously known constructions. Since we wish to get locally list-decodable codes with a constant query complexity, we need to use a locally decodable code with a constant query complexity. The best such codes currently known are due to [MFL$^+$10]:

Theorem 5 ([MFL$^+$10])**.** *For every $r \geq 2$ there exists a code*

$$\mathcal{C}_{\mathrm{LDC}} : \{0,1\}^n \mapsto \{0,1\}^{\exp(\exp(O(\sqrt[r]{\log n (\log\log n)^{r-1}})))}$$

which is $\left(\left(\frac{3}{4}\right)^{\min\{51,\lfloor r/2 \rfloor\}} 2^r, \gamma, 2 \cdot \left(\frac{3}{4}\right)^{\min\{51,\lfloor r/2 \rfloor\}} 2^r \cdot \gamma\right)$ locally decodable, for every $\gamma > 0$.

Let $\mathcal{C}_{\mathrm{LDC}}$ and $\mathcal{C}_{\mathrm{App}}$ be the codes from Theorem 5 and Theorem 2, respectively. Applying Theorem 4 on these codes gives:

Corollary 6 (Theorem 1 restated)**.** *For every $r \geq 2$ and every $\alpha, \epsilon > 0$ there exists a code*

$$\mathcal{C} : \{0,1\}^n \mapsto \{0,1\}^{f(\alpha, \frac{\epsilon}{2 \cdot 3^{r/2}}) \cdot \exp(\exp(O(\sqrt[r]{\log n (\log\log n)^{r-1}})))}$$

which is $\left(1/2 + \alpha, O(\frac{1}{\alpha^2}), O\left(\frac{r + \log(1/\epsilon)}{\alpha^3} \cdot 2^r\right), \epsilon\right)$ locally list-decodable, where f is the constant from Theorem 2.

Acknowledgements. We thank Or Meir and Sergey Yekhanin for useful comments

References

[AS03] Sanjeev Arora and Madhu Sudan. Improved low-degree testing and its applications. *Combinatorica*, 23(3):365–426, 2003.

[BET10] Avraham Ben-Aroya, Klim Efremenko, and Amnon Ta-Shma. Local list-decoding with a constant number of queries. In *FOCS*, 2010.

[DGY10] Zeev Dvir, Parikshit Gopalan, and Sergey Yekhanin. Matching vector codes. In *FOCS*, 2010.

[Efr09] Klim Efremenko. 3-query locally decodable codes of subexponential length. In *STOC*, pages 39–44, 2009.

[Eli57] Peter. Elias. List decoding for noisy channels. Technical report, Research Laboratory of Electronics, Massachusetts Institute of Technology, 1957.

[GKST02] Oded Goldreich, Howard J. Karloff, Leonard J. Schulman, and Luca Trevisan. Lower bounds for linear locally decodable codes and private information retrieval. In *IEEE Conference on Computational Complexity*, pages 175–183, 2002.

[GKZ08] Parikshit Gopalan, Adam R. Klivans, and David Zuckerman. List-decoding reed-muller codes over small fields. In *STOC*, pages 265–274, 2008.

[GL89] Oded Goldreich and Leonid A. Levin. A hard-core predicate for all one-way functions. In *STOC*, pages 25–32, 1989.

[GM09] Anna Gal and Andrew Mills. Three query locally decodable codes with higher correctness require exponential length. 2009.

[GRS00] Oded Goldreich, Ronitt Rubinfeld, and Madhu Sudan. Learning polynomials with queries: The highly noisy case. *SIAM J. Discrete Math.*, 13(4):535–570, 2000.

[IJK06] Russell Impagliazzo, Ragesh Jaiswal, and Valentine Kabanets. Approximately list-decoding direct product codes and uniform hardness amplification. In *FOCS*, pages 187–196, 2006.

[IJKW08] Russell Impagliazzo, Ragesh Jaiswal, Valentine Kabanets, and Avi Wigderson. Uniform direct product theorems: simplified, optimized, and derandomized. In *STOC*, pages 579–588, 2008.

[IJKW10] Russell Impagliazzo, Ragesh Jaiswal, Valentine Kabanets, and Avi Wigderson. Uniform direct product theorems: Simplified, optimized, and derandomized. *SIAM J. Comput.*, 39(4):1637–1665, 2010.

[IS08] Toshiya Itoh and Yasuhiro Suzuki. New constructions for query-efficient locally decodable codes of subexponential length. *CoRR*, abs/0810.4576, 2008.

[IW97] Russell Impagliazzo and Avi Wigderson. P=BPP unless E has subexponential circuits: derandomizing the XOR lemma. In *STOC*, pages 220–229, 1997.

[KdW03] Iordanis Kerenidis and Ronald de Wolf. Exponential lower bound for 2-query locally decodable codes via a quantum argument. In *STOC*, pages 106–115. ACM, 2003.

[KS07] Tali Kaufman and Madhu Sudan. Sparse random linear codes are locally decodable and testable. In *FOCS*, pages 590–600, 2007.

[KS09] Swastik Kopparty and Shubhangi Saraf. Local list-decoding and testing of sparse random linear codes from high-error. Technical Report 115, Electronic Colloquium on Computational Complexity (ECCC), 2009.

[KT00] Jonathan Katz and Luca Trevisan. On the efficiency of local decoding procedures for error-correcting codes. In *STOC*, pages 80–86, 2000.

[MFL+10] Y. Meng Chee, T. Feng, S. Ling, H. Wang, and L. F. Zhang. Query-Efficient Locally Decodable Codes of Subexponential Length. *ArXiv e-prints*, August 2010.

[Rag07] Prasad Raghavendra. A note on yekhanin's locally decodable codes. *Electronic Colloquium on Computational Complexity (ECCC)*, 2007.

[STV01] Madhu Sudan, Luca Trevisan, and Salil Vadhan. Pseudorandom generators without the XOR lemma. *Journal of Computer and System Sciences*, 62(2):236–266, 2001.

[Tre03] Luca Trevisan. List Decoding Using the XOR Lemma. In *FOCS*, pages 126–135, 2003.

[Woo07] David Woodruff. New lower bounds for general locally decodable codes. *Electronic Colloquium on Computational Complexity (ECCC)*, 2007.

[Woo08] David P. Woodruff. Corruption and recovery-efficient locally decodable codes. In *APPROX-RANDOM*, pages 584–595, 2008.

[Woz58] John M. Wozencraft. List decoding. *Quarterly progress report, Research Laboratory of Electronics, MIT*, 48:90–95, 1958.

[Yek08] Sergey Yekhanin. Towards 3-query locally decodable codes of subexponential length. *J. ACM*, 55(1), 2008.

ECCC ISSN 1433-8092

http://eccc.hpi-web.de