Electronic Colloquium on Computational Complexity, Report No. 68 (2015)

High rate locally-correctable and locally-testable codes with sub-polynomial query complexity*

Swastik Kopparty† Or Meir‡ Noga Ron-Zewi§ Shubhangi Saraf¶

April 21, 2015

Abstract

In this work, we construct the first locally-correctable codes (LCCs), and locally-testable codes (LTCs) with constant rate, constant relative distance, and sub-polynomial query complexity. Specifically, we show that there exist binary LCCs and LTCs with block length n, constant rate (which can even be taken arbitrarily close to 1), constant relative distance, and query complexity $\exp(\tilde{O}(\sqrt{\log n}))$. Previously such codes were known to exist only with $\Omega(n^{\beta})$ query complexity (for constant $\beta > 0$), and there were several, quite different, constructions known.

Our codes are based on a general distance-amplification method of Alon and Luby [AL96]. We show that this method interacts well with local correctors and testers, and obtain our main results by applying it to suitably constructed LCCs and LTCs in the non-standard regime of *sub-constant relative distance*.

Along the way, we also construct LCCs and LTCs over large alphabets, with the same query complexity $\exp(\tilde{O}(\sqrt{\log n}))$, which additionally have the property of approaching the Singleton bound: they have almost the best-possible relationship between their rate and distance. This has the surprising consequence that asking for a large alphabet error-correcting code to further be an LCC or LTC with $\exp(\tilde{O}(\sqrt{\log n}))$ query complexity does not require any sacrifice in terms of rate and distance! Such a result was previously not known for any $o(n)$ query complexity.

Our results on LCCs also immediately give locally-decodable codes (LDCs) with the same parameters.

*A preliminary version of this work appeared as [Mei14].

†Department of Mathematics & Department of Computer Science, Rutgers University, Piscataway NJ 08854, USA. Supported in part by a Sloan Fellowship and NSF grant CCF-1253886. swastik.kopparty@gmail.com

‡Department of Computer Science and Applied Mathematics, Weizmann Institute of Science, Rehovot 76100, Israel. This research was carried out when Meir was supported in part by the Israel Science Foundation (grant No. 460/05). or.meir@weizmann.ac.il

§School of Mathematics, Institute for Advanced Study, Princeton, NJ, USA. Supported in part by the Rothschild fellowship and NSF grant CCF-1412958. nogazewi@ias.edu

¶Department of Mathematics & Department of Computer Science, Rutgers University, Piscataway NJ 08854, USA. Supported in part by NSF grant CCF-1350572. shubhangi.saraf@gmail.com

ISSN 1433-8092

1 Introduction

Locally-correctable codes [BFLS91, STV01, KT00] and locally-testable codes [FS95, RS96, GS06] are codes that admit local algorithms for decoding and testing respectively. More specifically:

- We say that a code C is a locally-correctable code (LCC)[1] if there is a randomized algorithm that, when given a string z that is close to a codeword $c \in C$, and a coordinate i, computes c_i while making only a small number of queries to z.
- We say that a code C is a locally-testable code (LTC) if there is a randomized algorithm that, when given a string z, decides whether z is a codeword of C, or far from C, while making only a small number of queries to z.

The number of queries that are used by the latter algorithms is called the query complexity.

Besides being interesting in their own right, LCCs and LTCs have also played important roles in different areas of complexity theory, such as hardness amplification and derandomization (see e.g. [STV01]), and probabilistically checkable proofs [AS98, ALM$^+$98]. It is therefore a natural and well-known question to determine what are the best parameters that LCCs and LTCs can achieve.

LCCs and LTCs were originally studied in the setting where the query complexity was either constant or poly-logarithmic. In those settings, it is believed that LCCs and LTCs must be very redundant, since every bit of the codeword must contain, in some sense, information about every other bit of the codeword. Hence, we do not expect such codes to achieve a high rate. In particular, in the setting of constant query complexity, it is known that linear LCCs cannot have constant rate [KT00, WdW05, Woo07][2], and that LTCs with certain restrictions cannot have constant rate [DK11, BSV12]. On the other hand, the best-known constant-query LCCs have exponential length[3], and the best-known constant-query LTCs have quasi-linear length (see e.g. [BS08, Din07, Vid15]).

It turns out that the picture is completely different when allowing the query complexity to be much larger. In this setting, it has long been known that one can have LCCs and LTCs with constant rate and query complexity $O(n^{\beta})$ for constant $\beta > 0$ [BFLS91, RS96]. More recently, it has been discovered that both LCCs [KSY14, GKS13, HOW13] and LTCs [Vid11, GKS13] can simultaneously achieve rates that are arbitrarily close to 1 and query complexity $O(n^{\beta})$ for an arbitrary constant $\beta > 0$. This is in contrast with the general belief that local correctability and testability require much redundancy.

In this work, we show that there are LCCs and LTCs with constant rate (which can in fact be taken to be arbitrarily close to 1) and constant relative distance, whose associated local algorithms have $n^{o(1)}$ query complexity and running time. We find it quite surprising in light of the fact that there were several quite different constructions of LCCs and LTCs [BFLS91, RS96, KSY14, Vid11, GKS13, HOW13] with constant rate and constant relative distance, all of which had $\Omega(n^{\beta})$ query complexity.

Furthermore, we show that over large alphabets, such codes can approach the Singleton bound: they achieve a tradeoff between rate and distance which is essentially as good as possible for general

[1] There is a closely related notion of locally decodable codes (LDCs) that is more popular and very well studied. All our results for LCCs hold for LDCs as well, see discussion at the end of the introduction.

[2] [KT00, WdW05, Woo07] proved a lower bound for the related notion of LDCs. Since every linear LCC is also an LDC, their lower bound applies to linear LCCs as well.

[3] For example, a constant-degree Reed-Muller code is such an LCC.

error-correcting codes. Such a result was previously not known for any $o(n)$ query complexity. This means that, remarkably, local correctability and local testability with $n^{o(1)}$ queries over large alphabets is not only possible with constant rate and constant relative distance, but it also does not require "paying" anything in terms of rate and relative distance.

We first state our theorems for the binary alphabet.

Theorem 1.1 (Binary LCCs with sub-polynomial query complexity)**.** *For every $r \in (0,1)$, there exist $\delta > 0$ and an explicit infinite family of binary linear codes $\{C_n\}_n$ satisfying:*

1. *C_n has block length n, rate at least r, and relative distance at least δ,*
2. *C_n is locally correctable from $\frac{\delta}{2}$-fraction of errors with query complexity and running time at most* $\exp(\sqrt{\log n \cdot \log\log n})$.

Theorem 1.2 (Binary LTCs with sub-polynomial query complexity)**.** *For every $r \in (0,1)$, there exist $\delta > 0$ and an explicit infinite family of binary linear codes $\{C_n\}_n$ satisfying:*

1. *C_n has block length n, rate at least r, and relative distance at least δ,*
2. *C_n is locally testable with query complexity and running time at most* $\exp(\sqrt{\log n \cdot \log\log n})$.

The binary LCCs and LTCs in the above theorems are obtained by first constructing LCCs and LTCs over large alphabets, and then concatenating them with binary codes. The following theorems describe these large alphabet LCCs and LTCs, which in addition to having sub-polynomial query complexity, also approach the Singleton bound.

Theorem 1.3 (LCCs with sub-polynomial query complexity approaching the Singleton bound)**.** *For every $r \in (0,1)$, there exists an explicit infinite family of linear codes $\{C_n\}_n$ satisfying:*

1. *C_n has block length n, rate at least r, and relative distance at least* $1 - r - o(1)$,
2. *C_n is locally correctable from $\frac{1-r-o(1)}{2}$-fraction of errors with query complexity and running time at most* $\exp(\sqrt{\log n \cdot \log\log n})$,
3. *The alphabet of C_n is of size at most* $\exp(\exp(\sqrt{\log n \cdot \log\log n}))$.

Theorem 1.4 (LTCs with sub-polynomial query complexity approaching the Singleton bound)**.** *For every $r \in (0,1)$, there exists an explicit infinite family of linear codes $\{C_n\}_n$ satisfying:*

1. *C_n has block length n, rate at least r, and relative distance at least* $1 - r - o(1)$,
2. *C_n is locally testable with query complexity and running time at most* $\exp(\sqrt{\log n \cdot \log\log n})$,
3. *The alphabet of C_n is of size at most* $\exp(\exp(\sqrt{\log n \cdot \log\log n}))$.

The above theorems are proved in Sections 3 and 4.

Remark 1.5. If we were only interested in LCCs and LTCs with $O(n^\beta)$ query complexity (for arbitrary β), we could have constructed binary codes that meet the *Zyablov bound*, which is the best-known rate-distance tradeoff for explicit binary codes. Furthermore, we could have constructed codes over constant-size alphabet that approach the Singleton bound (rather than having alphabet of super-constant size).

Moreover, our results imply the existence of *non-explicit* binary LCCs/LTCs with query complexity $\exp(\sqrt{\log n \cdot \log\log n})$ that meet the Zyablov bound. This follows by concatenating the codes of Theorems 1.3 and 1.4 with (non-explicit) Gilbert-Varshamov codes [Gil52, Var57].

The Alon-Luby distance-amplification. Our constructions are based on the distance-amplification technique of [AL96]. This distance amplifier, based on a d-regular expander, converts an error-correcting code with relative distance $\gg 1/d$ into an error-correcting code with larger relative distance δ, while reducing the rate only by a factor of $\approx (1-\delta)$. Thus for a large enough constant d, if we start with a code of rate $1-\varepsilon$ and relative distance $\gg 1/d$, where $\varepsilon \ll \delta$, then after distance amplification with a d-regular expander, we get a code with rate $(1-\delta)(1-\varepsilon) \approx (1-\delta)$ and relative distance δ.

The original application of this technique in [AL96] was to construct linear-time erasure-decodable codes approaching the Singleton bound. In addition to the above distance-amplification technique, [AL96] constructed a linear-time erasure-decodable code (not approaching the Singleton bound) which could be used as the input code to the amplifier. The main result of [AL96] then follows from the fact that distance amplification via a *constant-degree* expander preserves linear-time erasure-decodability.

Subsequent applications of this distance-amplification technique followed a similar outline. One first constructs codes with high rate with some (possibly very small) constant relative distance and a certain desirable property. Then, applying distance amplification with a (possibly very large) constant-degree expander, one obtains a code with a much better tradeoff between its rate and relative distance. Finally one shows that the distance amplification with a constant degree expander preserves the desirable property. This scheme was implemented in [GI05], who constructed codes that can be decoded in linear time from *errors* (rather than *erasures*), and in [GI02, GR08], who constructed capacity-achieving list-decodable codes with constant alphabet.

Our observations. The first main observation of this paper is that the distance-amplification technique also preserves the property of being an LCC or an LTC. Specifically, if we start with an LCC or LTC with query complexity q, and then apply distance amplification with a d-regular expander, then the resulting code is an LCC/LTC with query complexity $q \cdot \mathrm{poly}(d)$.

The next main observation is that this connection continues to hold even if we take d to be super-constant, and take the LCC or LTC to have sub-constant relative distance $\Theta(1/d)$ (and then we only require the LCC to be able to correct strings whose distance from the code is within some constant fraction of the minimum distance of the code). This is potentially useful, since we only blow up the query complexity by a factor of $\mathrm{poly}(d)$, and perhaps LCCs/LTCs with high rate and sub-constant relative distance can have improved query complexity over their constant relative distance counterparts.

Finally, we show that existing families of high rate LCCs and LTCs can achieve sub-polynomial query complexity if we only require them to have sub-constant relative distance. Specifically, multiplicity codes [KSY14] in a super-constant number of variables give us the desired LCCs, and super-constant-wise tensor products [Vid11] give us the desired LTCs.

As far as we are aware, there have been no previous uses of this distance-amplification technique using an expander of super-constant degree.

More generally, we wish to draw attention to the technique of [AL96]. We believe that it should be viewed as a general scheme for improving the rate-distance tradeoff for codes with certain desirable properties. In particular, it may transfer properties that codes with constant rate and sub-constant relative distance are known to have, to codes with constant rate and constant relative distance, and even to codes approaching the Singleton bound. We believe that this is a good "take-home message" from this work.

Correctable *and* testable codes. Using the above method, it is also possible to construct improved codes that are simultaneously locally correctable and locally testable. This can be done by applying the distance-amplification technique to the lifted Reed-Solomon codes of [GKS13]. The codes of [GKS13] are both locally correctable and testable, and achieve rates that are arbitrarily close to 1. Using these codes of [GKS13] in the sub-constant relative distance regime, and combining with our framework, we get codes of constant rate and constant relative distance (which over large alphabets approach the Singleton bound) that are both locally correctable and locally testable with $n^{O(1/\log\log n)}$ queries.

Locally decodable codes. An important variant of LCCs are locally decodable codes (LDCs). Those codes are defined similarly to LCCs, with the following difference: Recall that in the definition of LCCs, the decoder gets access to a string z which is close to a codeword c, and is required to decode a coordinate of c. In the definition of LDCs, we view the codeword c as the encoding of some message x, and the decoder is required to decode a coordinate of x. LDCs were studied extensively in the literature, perhaps more so than LCCs (see [Yek12] for a survey). One notable fact about LDCs is that there are constructions of LDCs with a constant query complexity and sub-exponential length [Yek08, Rag07, KY09, Efr12].

If we restrict ourselves to *linear* codes, then LDCs are a weaker object than LCCs, since every linear LCC can be converted into an LDC by choosing a systematic encoding map[4]. Since the LCCs we construct in this paper are linear, all our results apply to LDCs as well.

Organization of this paper. We review the required preliminaries in Section 2, construct our LCCs in Section 3, and construct our LTCs in Section 4. We conclude with some open questions in Section 5.

Version. A preliminary version of this paper appeared as [Mei14], where the distance-amplification technique was used to construct codes approaching the Singleton bound with query complexity $O(n^\beta)$ (for arbitrary $\beta > 0$).

2 Preliminaries

All logarithms in this paper are in base 2. For any $n \in \mathbb{N}$ we denote $[n] \stackrel{\text{def}}{=} \{1 \ldots, n\}$. We denote by $\mathbb{F}_2$ the finite field of two elements. For any finite alphabet Σ and any pair of strings $x, y \in \Sigma^n$, the relative Hamming distance (or, simply, relative distance) between x and y is the fraction of coordinates on which x and y differ, and is denoted by $\mathrm{dist}(x, y) \stackrel{\text{def}}{=} |\{i \in [n] : x_i \neq y_i\}|/n$. We have the following useful approximation.

Fact 2.1. *For every $x, y \in \mathbb{R}$ such that $0 \leq x \cdot y \leq 1$, it holds that*

$$(1 - x)^y \leq 1 - \frac{1}{4} \cdot x \cdot y.$$

Proof. It holds that

$$(1 - x)^y \leq e^{-x \cdot y} \leq 1 - \frac{1}{4} \cdot x \cdot y.$$

[4]This conversion will lead to an LDC with the same query complexity, but the running time of the local decoder will be small only if the systematic encoding map can be computed efficiently.

The second inequality relies on the fact that $1 - \frac{1}{4} \cdot x \geq e^{-x}$ for every $x \in (0, 1)$, which can be proved by noting that $1 - \frac{1}{4} \cdot x = e^{-x}$ at $x = 0$, and that the derivative of e^{-x} is smaller than that of $1 - \frac{1}{4} \cdot x$ for every $x \in (0, 1)$. The first inequality relies on the fact that $1 - x \leq e^{-x}$ for every $x \in \mathbb{R}$, which can be proved using similar considerations. ∎

2.1 Error correcting codes

Let Σ be an alphabet and let n be a positive integer (the block length). A code is simply a subset $C \subseteq \Sigma^n$. If $\mathbb{F}$ is a finite field and Σ is a vector space over $\mathbb{F}$, we say that a code $C \subseteq \Sigma^n$ is $\mathbb{F}$-*linear* if it is an $\mathbb{F}$-linear subspace of the $\mathbb{F}$-vector space Σ^n. The rate of a code is the ratio $\frac{\log|C|}{\log(|\Sigma|^n)}$, which for $\mathbb{F}$-linear codes equals $\frac{\dim_{\mathbb{F}}(C)}{n \cdot \dim_{\mathbb{F}}(\Sigma)}$.

The elements of a code C are called codewords. We say that C has relative distance at least δ if for every pair of distinct codewords $c_1, c_2 \in C$ it holds that $\mathrm{dist}(c_1, c_2) \geq \delta$. We will use the notation $\mathrm{dist}(w, C)$ to denote the relative distance of a string $w \in \Sigma^n$ from C, and say that w is ε-close (respectively, ε-far) to C if $\mathrm{dist}(w, C) < \varepsilon$ (respectively, if $\mathrm{dist}(w, C) \geq \varepsilon$).

An encoding map for C is a bijection $E_C : \Sigma^k \to C$, where $|\Sigma|^k = |C|$. We say that an infinite family of codes $\{C_n\}_n$ is explicit if there is a polynomial time algorithm that computes the encoding maps of all the codes in the family. For a code C of relative distance δ, a given parameter $\tau < \delta/2$, and a string $z \in \Sigma^n$, the problem of decoding from τ fraction of errors is the task of finding the unique $c \in C$ (if any) which satisfies $\mathrm{dist}(c, z) \leq \tau$.

Reed-Solomon codes. We use the following fact, which states the existence of Reed-Solomon codes and their relevant properties.

Fact 2.2 (Reed-Solomon Codes [RS60])**.** *For every $k, n \in \mathbb{N}$ such that $n \geq k$, and for every finite field $\mathbb{F}$ such that $|\mathbb{F}| \geq n$, there exists an $\mathbb{F}$-linear code $RS_{k,n} \subseteq \mathbb{F}^n$ with rate $r = k/n$, and relative distance at least $1 - \frac{k-1}{n} > 1 - r$. Furthermore, $RS_{k,n}$ has an encoding map $E : \mathbb{F}^k \to RS_{k,n}$ which can be computed in time $\mathrm{poly}(n, \log|\mathbb{F}|)$, and can be decoded from up to $(1 - \frac{k-1}{n})/2$ fraction of errors in time $\mathrm{poly}(n, \log|\mathbb{F}|)$.*

2.2 Locally-correctable codes

Intuitively, a code is said to be locally correctable [BFLS91, STV01, KT00] if, given a codeword $c \in C$ that has been corrupted by some errors, it is possible to decode any coordinate of c by reading only a small part of the corrupted version of c. Formally, it is defined as follows.

Definition 2.3. We say that a code $C \subseteq \Sigma^n$ is locally correctable from τ-fraction of errors with query complexity q if there exists a randomized algorithm A that satisfies the following requirements:

- **Input:** A takes as input a coordinate $i \in [n]$ and also gets oracle access to a string $z \in \Sigma^n$ that is τ-close to a codeword $c \in C$.
- **Output:** A outputs c_i with probability at least $\frac{2}{3}$.
- **Query complexity:** A makes at most q queries to the oracle z.

We say that the algorithm A is a local corrector of C. Given an infinite family of LCCs $\{C_n\}_n$, a uniform local corrector for the family is a randomized oracle algorithm that given n, computes the local corrector of C_n. We will often be also interested in the running time of the uniform local corrector.

Remark 2.4. The above success probability of $\frac{2}{3}$ can be amplified using sequential repetition, at the cost of increasing the query complexity. Specifically, amplifying the success probability to $1 - e^{-t}$ requires increasing the query complexity by a factor of $O(t)$.

2.3 Locally-testable codes

Intuitively, a code is said to be locally testable [FS95, RS96, GS00] if, given a string $z \in \Sigma^n$, it is possible to determine whether z is a codeword of C, or rather far from C, by reading only a small part of z. There are two variants of LTCs in the literature, "weak" LTCs and "strong" LTCs. From now on, we will work exclusively with strong LTCs, since it is a simpler notion and allows us to state a stronger result.

Definition 2.5. We say that a code $C \subseteq \Sigma^n$ is (strongly) locally testable with query complexity q if there exists a randomized algorithm A that satisfies the following requirements:

- **Input:** A gets oracle access to a string $z \in \Sigma^n$.
- **Completeness:** If z is a codeword of C, then A accepts with probability 1.
- **Soundness:** If z is not a codeword of C, then A rejects with probability at least $\mathrm{dist}(z, C)$.
- **Query complexity:** A makes at most q *non-adaptive* queries to the oracle z.

We say that the algorithm A is a local tester of C. Given an infinite family of LTCs $\{C_n\}_n$, a uniform local tester for the family is a randomized oracle algorithm that given n, computes the local tester of C_n. Again, we will often also be interested in the running time of the uniform local tester.

A remark on amplifying the rejection probability. It is common to define strong LTCs with an additional parameter ρ, and have the following soundness requirement:

- If z is not a codeword of C, then A rejects with probability at least $\rho{\cdot}\mathrm{dist}(z, C)$.

Our definition corresponds to the special case where $\rho = 1$. However, given an LTC with $\rho < 1$, it is possible to amplify ρ up to 1 at the cost of increasing the query complexity. Hence, we chose to fix ρ to 1 in our definition, which somewhat simplifies the presentation.

The amplification of ρ is performed as follows: The amplified tester invokes the original tester A for $\frac{4}{\rho}$ times, and accepts only if all invocations of A accept. Clearly, this increases the query complexity by a factor of $\frac{4}{\rho}$ and preserves the completeness property. To analyze the rejection probability, let z be a string that is not a codeword of C, and observe that the amplified tester rejects it with probability at least

$$\begin{aligned} & 1 - (1 - \rho \cdot \mathrm{dist}(z, C))^{\frac{4}{\rho}} \\ \geq \;& 1 - \left(1 - \frac{1}{4} \cdot \frac{4}{\rho} \cdot \rho \cdot \mathrm{dist}(z, C)\right) \qquad \text{(Fact 2.1)} \\ = \;& \mathrm{dist}(z, C), \end{aligned}$$

as required.

2.4 Expander graphs

Expander graphs are graphs with certain pseudorandom connectivity properties. Below, we state the construction and properties that we need. The reader is referred to [HLW06] for a survey. For a graph G, a vertex s and a set of vertices T, let $E(s,T)$ denote the set of edges that go from s into T.

Definition 2.6. Let $G = (U \cup V, E)$ be a bipartite d-regular graph with $|U| = |V| = n$. We say that G is an (α, γ)-$\mathsf{sampler}$ if the following holds for every $T \subseteq V$: For at least $1 - \alpha$ fraction of the vertices $s \in U$ it holds that

$$\frac{|E(s,T)|}{d} - \frac{|T|}{n} \leq \gamma.$$

Lemma 2.7. *For every $\alpha, \gamma > 0$ and every sufficiently large $n \in \mathbb{N}$ there exists a bipartite d-regular graph $G_{n,\alpha,\gamma} = (U \cup V, E)$ with $|U| = |V| = n$ and $d = \mathrm{poly}\left(\frac{1}{\alpha \cdot \gamma}\right)$ such that $G_{n,\alpha,\gamma}$ is an (α, γ)-sampler. Furthermore, there exists an algorithm that takes as inputs n, α, γ, and a vertex w of $G_{n,\alpha,\gamma}$, and computes the list of the neighbors of w in $G_{n,\alpha,\gamma}$ in time* $\mathrm{poly}(\frac{\log n}{\alpha \cdot \gamma})$.

Proof sketch. A full proof of Lemma 2.7 requires several definitions and lemmas that we have not stated, such as second eigenvalue, edge expansion, and the expander mixing lemma. Since this is not the focus of this paper, we only sketch the proof without stating those notions. The interested reader is referred to [HLW06].

Let α, γ and n be as in the lemma. We sketch the construction of the graph $G \stackrel{\mathrm{def}}{=} G_{n,\alpha,\gamma}$. First, observe that it suffices to construct a strongly-explicit *non-bipartite* graph G' over n vertices (that is, a graph G' in which the neighborhood of any given vertex is computable in time $\mathrm{poly}(\log n)$) with the desired property. The reason is that each such graph G' can be converted into a bipartite graph G with the desired property, by taking two copies of the vertex set of G' and connecting the two copies according to the edges in G'. The existence of the algorithm stated in the lemma follows from the fact that G' is strongly-explicit.

We thus focus on constructing the graph G'. This is done in two steps: first, we show how to construct a strongly-explicit expander G'' over n vertices – this requires a bit of work, since n can be an arbitrary number, and expanders are usually constructed for special values of n. In the second step, we amplify the spectral gap of G'' by powering, and set G' to be the powered graph. We then prove that G' has the desired sampling property.

The first step. The work of [GG81] gives a strongly-explicit expander with constant degree and constant edge expansion for every n that is a square, so we only need to deal with the case in which n is not a square. Suppose that $n = m^2 - k$, where m^2 is the minimal square larger than n, and observe that $k \leq 2m - 1$, which is at most $\frac{1}{2} \cdot m^2$ for sufficiently large m. Now, we construct an expander over m^2 vertices using [GG81], and then merge k pairs of vertices. In order to maintain the regularity, we add self-loops to all the vertices that were not merged. We set G'' to be the resulting graph.

It is easy to see that G'' is a regular graph over n vertices. Since the merge and the addition of self-loops maintain the degree and the edge expansion of the original expander up to a constant factor, it follows that G'' is an expander with constant degree and constant edge expansion. Furthermore, it is not hard to see that G'' is strongly-explicit.

The second step. Since G'' is an expander, and in particular has constant edge expansion, it follows from the Cheeger inequality [Dod84, AM85] that its second-largest normalized eigenvalue (in absolute value) is some constant smaller than 1. Let us denote this normalized eigenvalue by λ. We note that the degree and the edge expansion of G'', as well as λ, are independent of n.

We now construct the graph G' by raising G'' to the power $\log_\lambda(\sqrt{\alpha} \cdot \gamma)$. Observe that G' is a graph over n vertices with degree $d \stackrel{\text{def}}{=} \mathrm{poly}\left(\frac{1}{\alpha \cdot \gamma}\right)$ and normalized second eigenvalue $\sqrt{\alpha} \cdot \gamma$. It is not hard to see that G' is strongly-explicit.

The sampling property. We prove that G' has the desired sampling property. Let T be a subset of vertices of G'. We show that for at least $(1-\alpha)$ fraction of the vertices s of G' it holds that

$$\frac{|E(s,T)|}{d} - \frac{|T|}{n} \leq \gamma.$$

To this end, let

$$S \stackrel{\text{def}}{=} \left\{ s \in U \;\middle|\; \frac{|E(s,T)|}{d} - \frac{|T|}{n} > \gamma \right\}.$$

Clearly, it holds that

$$\frac{|E(S,T)|}{d \cdot |S|} - \frac{|T|}{n} > \gamma.$$

On the other hand, the expander mixing lemma [AC88] implies that

$$\frac{|E(S,T)|}{d \cdot |S|} - \frac{|T|}{n} \leq \sqrt{\alpha} \cdot \gamma \cdot \sqrt{\frac{|T|}{|S|}}.$$

By combining the above pair of inequalities, we get

$$\gamma < \sqrt{\alpha} \cdot \gamma \cdot \sqrt{\frac{|T|}{|S|}}$$

$$|S| < \alpha \cdot |T| \leq \alpha \cdot n,$$

as required. ■

3 LCCs with sub-polynomial query complexity

In this section, we prove the following theorem on LCCs, which immediately implies Theorem 1.3 from the introduction.

Theorem 3.1 (Main LCC theorem)**.** *For every $r \in (0,1)$, there exists an explicit infinite family of $\mathbb{F}_2$-linear codes $\{C_n\}_n$ satisfying:*

1. *C_n has block length n, rate at least r, and relative distance at least $1 - r - o(1)$.*
2. *C_n is locally correctable from $\frac{1-r-o(1)}{2}$ fraction of errors with query complexity $\exp(\sqrt{\log n \cdot \log\log n})$.*
3. *The alphabet of C_n is a vector space Σ_n over $\mathbb{F}_2$, such that $|\Sigma_n| \leq \exp\left(\exp(\sqrt{\log n \cdot \log\log n})\right)$.*

Furthermore, the family $\{C_n\}_n$ *has a uniform local corrector that runs in time* $\exp(\sqrt{\log n \cdot \log\log n})$.

We note that the existence of binary LCCs (Theorem 1.1) also follows from Theorem 3.1: In order to construct the binary LCCs, we concatenate the codes of Theorem 3.1 with any asymptotically good inner binary code that has efficient encoding and decoding algorithms. The local corrector of the binary LCCs will emulate the original local corrector, and whenever the latter queries a symbol, the binary local corrector will emulate this query by decoding the corresponding codeword of the inner code. Since such constructions are standard (see [KSY14]), we do not provide the full details.

The proof of Theorem 3.1 has two steps. In the first step, we give a transformation that amplifies the fraction of errors from which an LCC can be corrected – this step follows the distance amplification of [AL96]. In the second step, we construct a locally-correctable code W_n with the the desired query complexity but that can only be corrected from a sub-constant fraction of errors. Finally, we construct the code C_n by applying the distance amplification to W_n. Those two steps are formalized in the following pair of lemmas, which are proved in Sections 3.1 and 3.2 respectively.

Lemma 3.2. *Suppose that there exists a code* W *that is locally correctable from* τ_W *fraction of errors with query complexity* q*, such that:*

- W *has rate* r_W.
- W *is* $\mathbb{F}_2$*-linear*

Then, for every $0 < \tau < \frac{1}{2}$ *and* $0 < \varepsilon < 1$*, there exists a code* C *that is locally correctable from* τ *fraction of errors with query complexity* $q \cdot \mathrm{poly}(1/(\varepsilon \cdot \tau_W))$*, such that:*

- $|C| = |W|$.
- C *has relative distance at least* $2 \cdot \tau$*, and rate at least* $r_W \cdot (1 - 2 \cdot \tau - \varepsilon)$.
- *Let* Λ *denote the alphabet of* W*. Then, the alphabet of* C *is* $\Sigma \stackrel{\text{def}}{=} \Lambda^p$ *for some* $p = \mathrm{poly}(1/(\varepsilon \cdot \tau_W))$.
- C *is* $\mathbb{F}_2$*-linear.*

Furthermore,

- *There is a polynomial time algorithm that computes a bijection from every code* W *to the corresponding code* C*, given* r_W*,* τ_W*,* r*,* ε *and* Λ.
- *There is an oracle algorithm that when given black box access to the local corrector of any code* W*, and given also* r_W*,* τ_W*,* r*,* ε*,* Λ*, computes the local corrector of the corresponding code* C*. The resulting local corrector of* C *runs in time that is polynomial in the running time of the local corrector of* W *and in* $1/\tau_W$*,* $1/\varepsilon$ *and* $\log(n_W)$ *where* n_W *is the block length of* W.

Lemma 3.3. *There exists an explicit infinite family of* $\mathbb{F}_2$*-linear codes* $\{W_n\}_n$ *satisfying:*

1. W_n *has block length* n*, rate at least* $1 - \frac{1}{\log n}$*, and relative distance at least* $\Omega\left(\sqrt{\frac{\log\log n}{\log^3 n}}\right)$.
2. W_n *is locally correctable from* $\Omega\left(\sqrt{\frac{\log\log n}{\log^3 n}}\right)$ *fraction of errors with query complexity* $\exp(\sqrt{\log n \cdot \log\log n})$.

3. The alphabet of W_n is a vector space Λ_n over $\mathbb{F}_2$, such that $|\Lambda_n| \le \exp\left(\exp(\sqrt{\log n \cdot \log\log n})\right)$.

Furthermore, the family $\{W_n\}_n$ has a uniform local corrector that runs in time $\exp(\sqrt{\log n \cdot \log\log n})$.

Proof of Theorem 3.1. We construct the family $\{C_n\}_n$ by applying Lemma 3.2 to the family $\{W_n\}_n$ of Lemma 3.3 with $\tau_W = \Omega\left(\sqrt{\frac{\log\log n}{\log^3 n}}\right)$, $\varepsilon = \frac{1}{\log n}$, and

$$\tau = \frac{1}{2} \cdot \left(1 - \frac{r}{1 - \frac{1}{\log n}} - \varepsilon\right) = \frac{1}{2} \cdot \left(1 - r - O\left(\frac{1}{\log n}\right)\right).$$

It is easy to see that C_n has the required rate, relative distance and alphabet size, and that it can be locally corrected from the required fraction of errors with the required query complexity. The family $\{C_n\}_n$ is explicit with the required running time due to the first item in the "furthermore" part of Lemma 3.2, and has a uniform local corrector due to the second item of that part. ■

Remark 3.4. In Lemma 3.2 above, we chose to assume that W is $\mathbb{F}_2$-linear for simplicity. More generally, if W is $\mathbb{F}$-linear for any finite field $\mathbb{F}$, then C is $\mathbb{F}$-linear as well. Furthermore, the lemma also works if W is not $\mathbb{F}$-linear for any field $\mathbb{F}$, in which case C is not guaranteed to be $\mathbb{F}$-linear for any field $\mathbb{F}$.

3.1 Proof of Lemma 3.2

3.1.1 Overview

Let $0 < \tau < \frac{1}{2}$. Our goal is to construct a code C that can be locally corrected from a fraction of errors at most τ. The idea of the construction is to combine the LCC W with a Reed-Solomon code to obtain a code C that enjoys "the best of both worlds": both the local correctability of W and the good error correction capability of Reed-Solomon. We do it in two steps: first, we construct a code C' which can be corrected from τ fraction of *random* errors. Then, we augment C' to obtain a code C that can be corrected from τ fraction of *adversarial* errors.

We first describe the construction of C'. To this end, we describe a bijection from W to C'. Let w be a codeword of W. To obtain the codeword $c' \in C'$ that corresponds to w, we partition w into blocks of length b (to be determined later), and encode each block with a Reed-Solomon code $RS_{b,d}$. We choose the relative distance of $RS_{b,d}$ to be $2 \cdot \tau + \varepsilon$, so its rate is $1 - 2 \cdot \tau - \varepsilon$ and the rate of C' is indeed $r_W \cdot (1 - 2 \cdot \tau - \varepsilon)$, as required.

We now claim that if one applies to a codeword $c' \in C'$ a noise that corrupts each coordinate with probability τ, then the codeword c' can be recovered from its corrupted version with high probability. To see it, first observe that with high probability, almost all the blocks of c' have at most $\tau + \frac{\varepsilon}{2}$ fraction of corrupted coordinates. Let us call those blocks "good blocks", and observe that the good blocks can be corrected by decoding them to the nearest codeword of $RS_{b,d}$ (since $\tau + \frac{\varepsilon}{2}$ is half the relative distance of $RS_{b,d}$). Next, observe that if b is sufficiently large, the fraction of "good blocks" is at least $1 - \tau_W$, and hence we can correct the remaining τ_W fraction of errors using the decoding algorithm of W. It follows that C' can be corrected from τ fraction of random errors, as we wanted.

Next, we show how to augment C' to obtain a code C that is correctable from adversarial errors. This requires two additional ideas. The first idea to apply a permutation that is "pseudorandom" in some sense to the coordinates of C'. The "pseudorandom" permutation is determined by the edges of an expander graph (see Section 2.4). This step is motivated by the hope that, after the adversary decided which coordinates to corrupt, applying the permutation to the coordinates will make the errors behave pseudorandomly. This will allow the above analysis for the case of random errors to go through.

Of course, on its own, this idea is doomed to fail, since the adversary can take the permutation into account when he chooses where to place the errors. Here the second idea comes into play: after applying the permutation to the coordinates of C', we will increase the alphabet size of the code, packing each block of symbols into a new big symbol. The motivation for this step is that increasing the alphabet size restricts the freedom of the adversary in choosing the pattern of errors. Indeed, we will show that after the alphabet size is increased, applying the permutation to the coordinates of the code makes the errors behave pseudorandomly. This allows us to prove that the code can be decoded from τ fraction of errors, as we wanted.

3.1.2 The construction of C

Choosing the parameters. Let W, r_W, τ_W, r, ε, and Λ be as in Lemma 3.2. Let $\{G_n\}_n$ be an infinite family of $(\tau_W, \frac{1}{2}\cdot\varepsilon)$-samplers as in Theorem 2.7, and let d be their degree.

Recall that we assumed that W is $\mathbb{F}_2$-linear, so $|\Lambda|$ is a power of 2. Let $\mathbb{F}$ be an extension field of $\mathbb{F}_2$, whose size is the minimal power of $|\Lambda|$ that is at least d. Let $RS_{b,d}$ be a Reed-Solomon code over $\mathbb{F}$ with relative distance $2\cdot\tau+\varepsilon$, rate $1-2\cdot\tau-\varepsilon$, and block length d.

Let n_W be the block length of W, and let t be such that $|\mathbb{F}| = |\Lambda|^t$. The block length of C will be $n \stackrel{\text{def}}{=} \frac{n_W}{b\cdot t}$, and its alphabet will be $\Sigma \stackrel{\text{def}}{=} \mathbb{F}^d$. Here, we assume that n_W is divisible by $b\cdot t$. If n_W is not divisible by $b\cdot t$, we consider two cases:

- if $n_W > b\cdot t/\varepsilon$, we increase n_W to the next multiple of $b\cdot t$ by padding the codewords of W with additional zero coordinates. This decreases the rate of W by at most ε, which essentially does not affect our results.

- Otherwise, we set C to be any Reed-Solomon code with blocklength n_W, relative distance $2\cdot\tau$, and rate $1-2\cdot\tau$. Observe that such a Reed-Solomon is locally correctable from τ fraction of errors with query complexity

$$n_W \le b\cdot t/\varepsilon = \mathrm{poly}(1/(\varepsilon\cdot\tau_W)),$$

  which satisfies our requirements.

A bijection from W to C. We construct the code C by describing a bijection from W to C. Given a codeword $w \in W$, one obtains the corresponding codeword $c \in C$ as follows:

- Partition w into $n \stackrel{\text{def}}{=} \frac{n_W}{b\cdot t}$ blocks of length $b\cdot t$. We view each of those blocks as a vector in $\mathbb{F}^b$, and encode it via the code $RS_{b,d}$. Let us denote the resulting string by $c' \in \mathbb{F}^{n\cdot d}$ and the resulting codewords of $RS_{b,d}$ by $B_1, \ldots, B_n \in \mathbb{F}^d$.

- Next, we apply a "pseudorandom" permutation to the coordinates of c' as follows: Let G_n be the graph from the infinite family above and let $U = \{u_1, \ldots, u_n\}$ and $V = \{v_1, \ldots, v_n\}$ be the left and right vertices of G_n respectively. For each $i \in [n]$ and $j \in [d]$, we write the j-th symbol of B_i on the j-th edge of u_i. Then, we construct new blocks $S_1, \ldots, S_n \in \mathbb{F}^d$, by setting the j-th symbol of S_i to be the symbol written on the j-th edge of v_i.

- Finally, we define the codeword c of $C \subseteq \Sigma^n$ as follows: the i-th coordinate c_i is the block S_i, reinterpreted as a symbol of the alphabet $\Sigma \stackrel{\text{def}}{=} \mathbb{F}^d$. We choose c to be the codeword in C that corresponds to the codeword w in W.

This concludes the definition of the bijection. It is not hard to see that this bijection can be computed in polynomial time, and that the code C is $\mathbb{F}_2$-linear. Furthermore, $\Sigma = \mathbb{F}^d = \Lambda^{t\cdot d}$ where $d \cdot t \leq d \log d = \operatorname{poly}(1/(\varepsilon \cdot \tau_W))$. The rate of C is

$$\begin{aligned}
\frac{\log |C|}{n \cdot \log |\Sigma|} &= \frac{\log |W|}{n \cdot d \cdot \log |\mathbb{F}|} \\
&= \frac{r_W \cdot \log |\Lambda^{n_W}|}{n \cdot d \cdot \log |\mathbb{F}|} \\
&= r_W \cdot \frac{n_W}{n} \cdot \frac{1}{d} \cdot \frac{\log |\Lambda|}{\log |\mathbb{F}|} \\
&= r_W \cdot (b \cdot t) \cdot \frac{1 - 2 \cdot \tau - \varepsilon}{b} \cdot \frac{1}{t} \\
&= r_W \cdot (1 - 2 \cdot \tau - \varepsilon),
\end{aligned}$$

as required. The relative distance of C is at least $2 \cdot \tau$ – although this could be proved directly, it also follows immediately from the fact that C is locally correctable from τ fraction of errors, which is proved in the next section.

3.1.3 Local correctability

In this section, we complete the proof of Lemma 3.2 by proving that C is locally correctable from τ fraction of errors with query complexity $\operatorname{poly}(d) \cdot q$. To this end, we describe a local corrector A. The algorithm A is based on the following algorithm A_0, which locally corrects coordinates of W from a corrupted codeword of C.

Lemma 3.5. *There exists an algorithm A_0 that satisfies the following requirements:*

- ***Input:*** *A_0 takes as input a coordinate $i \in [n_W]$, and also gets oracle access to a string $z \in \Sigma^n$ that is τ-close to a codeword $c \in C$.*
- ***Output:*** *Let w^c be the codeword of W from which c was generated. Then, A_0 outputs w^c_i with probability at least $1 - \frac{1}{3 \cdot b \cdot t \cdot d}$.*
- ***Query complexity:*** *A_0 makes $\operatorname{poly}(d) \cdot q$ queries to the oracle z.*

Before proving Lemma 3.5, we show how to construct the algorithm A given the algorithm A_0. Suppose that the algorithm A is given oracle access to a string z that is τ-close to a codeword $c \in C$, and a coordinate $i \in [n]$. The algorithm is required to decode c_i. Let $w^c \in \Lambda^{n_W}$ be the codeword of W from which c was generated, and let $B^c_1, \ldots, B^c_n$ and $S^c_1, \ldots, S^c_n$ be the corresponding blocks.

In order to decode c_i, the algorithm A should decode each of the symbols in the block $S^c_i \in \mathbb{F}^d$. Let $u_{j_1}, \ldots, u_{j_d}$ be the neighbors of v_i in the graph G_n. Each symbol of the block S^c_i belongs to one of the blocks $B^c_{j_1}, \ldots, B^c_{j_d}$, and therefore it suffices to retrieve the latter blocks. Now, each block $B^c_{j_h}$ is the encoding via $RS_{b,d}$ of $b \cdot t$ symbols of w^c (in the alphabet Λ). The algorithm A invokes the algorithm A_0 to decode each of those $b \cdot t$ symbols of w^c, for each of the blocks $B^c_{j_1}, \ldots, B^c_{j_d}$. By the union bound, the algorithm A_0 decodes all those $b \cdot t \cdot d$ symbols of w^c correctly with probability at least $1 - b \cdot t \cdot d \cdot \frac{1}{3 \cdot b \cdot t \cdot d} = \frac{2}{3}$. Whenever that happens, the algorithm A retrieves the blocks $B^c_{j_1}, \ldots, B^c_{j_d}$ correctly, and therefore computes the block S^c_i correctly. This concludes the construction of the algorithm A. Note that the query complexity of A is larger than that of A_0 by a factor of at most $b \cdot t \cdot d$, and hence it is at most $\mathrm{poly}(d) \cdot q$. It remains to prove Lemma 3.5.

Proof of Lemma 3.5. Let A_W be the local corrector of the code W. By amplification, we may assume that A_W errs with probability at most $\frac{1}{3 \cdot b \cdot t \cdot d}$, and this incurs a factor of at most $\mathrm{poly}(d)$ to its query complexity.

Suppose that the algorithm A_0 is invoked on a string $z \in \Sigma^n$ and a coordinate $i \in [n_W]$. The algorithm A_0 invokes the algorithm A_W to retrieve the coordinate i, and emulates A_W in the natural way: Recall that A_W expects to be given access to a corrupted codeword of W, and makes queries to it. Whenever A_W makes a query to a coordinate $i_W \in [n_W]$, the algorithm A_0 performs the following steps.

1. A_0 finds the block B_l to which the coordinate i_W belongs. Formally, $l \stackrel{\text{def}}{=} \lceil i_W/(b \cdot t) \rceil$.
2. A_0 finds the neighbors of the vertex u_l in G_n. Let us denote those vertices by $v_{j_1}, \ldots, v_{j_d}$.
3. A_0 queries the coordinates $j_1, \ldots j_d$, thus obtaining the blocks $S_{j_1}, \ldots, S_{j_d}$.
4. A_0 reconstructs the block B_l by reversing the permutation of G_n on $S_{j_1}, \ldots, S_{j_d}$.
5. A_0 attempts to decode B_l by applying an efficient decoding algorithm of Reed-Solomon.
6. Suppose that the decoding succeeded and returned a codeword of $RS_{b,d}$ that is $\left(\tau + \frac{\varepsilon}{2}\right)$-close to B_l. Then, A_0 retrieves the value of the i_W-th coordinate of w^c from the latter codeword, and feeds it to A_W as an answer to its query.
7. Otherwise, A_0 feeds 0 as an answer to the query of A_W.

When the algorithm A_W finishes running, the algorithm A_0 finishes and returns the output of A_W. It is not hard to see that the query complexity of A_0 is at most d times the query complexity of A_W, and hence it is at most $\mathrm{poly}(d) \cdot q$. It remains to show that A_0 succeeds in decoding from τ fraction of errors with probability at least $1 - \frac{1}{3 \cdot b \cdot t \cdot d}$.

Let $z \in \Sigma^n$ be a string that is τ-close to a codeword $c \in C$. Let $w^c \in \Lambda^{n_W}$ be the codeword of W from which c was generated, and let $B^c_1, \ldots, B^c_n$ and $S^c_1, \ldots, S^c_n$ be the corresponding blocks. We also use the following definitions:

1. Let $S^z_1, \ldots, S^z_n \in \mathbb{F}^d$ be the blocks that correspond to the symbols of z.
2. Let $B^z_1, \ldots, B^z_n$ be the blocks that are obtained from $S^z_1, \ldots, S^z_n$ by reversing the permutation.
3. Define blocks $B^{z\prime}_1, \ldots, B^{z\prime}_n$ as follows: if B^z_i is $\left(\tau + \frac{\varepsilon}{2}\right)$-close to $RS_{b,d}$, then $B^{z\prime}_i$ is the nearest codeword of $RS_{b,d}$. Otherwise, $B^{z\prime}_i$ is the all-zeroes block.
4. Let $w^z \in \Lambda^{n_W}$ be the string that is obtained by extracting the coordinates of w from each of the codewords $B^{z\prime}_1, \ldots, B^{z\prime}_n$.

It is easy to see that A_0 emulates the action of A_W on w^z. Therefore, if we prove that w^z is τ_W-close to w^c, we will be done. In order to do so, it suffices to prove that for at least $1-\tau_W$ fraction of the blocks B^z_l, it holds that B^z_l is $\left(\tau+\frac{\varepsilon}{2}\right)$-close to B^c_l.

To this end, let J be the set of coordinates on which z and c differ. In other words, for every $j \in J$ it holds that $S^z_j \neq S^c_j$. By assumption, $|J| \leq \tau \cdot n$. Now, observe that since G_n is a $\left(\tau_W, \frac{1}{2}\cdot\varepsilon\right)$-sampler, it holds that for at least $(1-\tau_W)$ fraction of the vertices u_l of G_n, there are at most $\left(\tau+\frac{\varepsilon}{2}\right)\cdot d$ edges between u_l and J. For each such u_l, it holds that $B^z_{u_l}$ is $\left(\tau+\frac{\varepsilon}{2}\right)$-close to $B^c_{u_l}$, and this concludes the proof. ■

It can be verified that the local correctors A_0 and A can be implemented efficiently with black box access to A_W, as required by the second item in the "furthermore" part of the lemma.

3.2 Proof of Lemma 3.3

In this section we prove Lemma 3.3, restated below.

Lemma 3.3. *There exists an explicit infinite family of $\mathbb{F}_2$-linear codes $\{W_n\}_n$ satisfying:*

1. *W_n has block length n, rate at least $1-\frac{1}{\log n}$, and relative distance at least $\Omega\left(\sqrt{\frac{\log\log n}{\log^3 n}}\right)$.*
2. *W_n is locally correctable from $\Omega\left(\sqrt{\frac{\log\log n}{\log^3 n}}\right)$ fraction of errors with query complexity $\exp(\sqrt{\log n\cdot\log\log n})$.*
3. *The alphabet of W_n is a vector space Λ_n over $\mathbb{F}_2$, such that $|\Lambda_n| \leq \exp\left(\exp(\sqrt{\log n\cdot\log\log n})\right)$.*

Furthermore, the family $\{W_n\}_n$ has a uniform local corrector that runs in time $\exp(\sqrt{\log n\cdot\log\log n})$.

For the proof of Lemma 3.3 we use the multiplicity codes of [KSY14], in a specialized sub-constant relative distance regime.

Lemma 3.6 ([KSY14, Lemma 3.5])**.** *Let $\mathbb{F}$ be any finite field. Let s, d, m be positive integers. Let M be the multiplicity code of order s evaluations of degree d polynomials in m variables over $\mathbb{F}$. Then M has block length $|\mathbb{F}|^m$, relative distance at least $\delta \stackrel{\text{def}}{=} 1-\frac{d}{s\cdot|\mathbb{F}|}$ and rate $\frac{\binom{d+m}{m}}{\binom{s+m-1}{m}\cdot|\mathbb{F}|^m}$, which is at least*

$$\left(\frac{s}{m+s}\right)^m\cdot\left(\frac{d}{s\cdot|\mathbb{F}|}\right)^m \geq \left(1-\frac{m^2}{s}\right)\cdot(1-\delta)^m.$$

The alphabet of C is $\mathbb{F}^{\binom{m+s-1}{m}}$, and C is $\mathbb{F}$-linear. Furthermore, there is $\operatorname{poly}\left(\mathbb{F}^m, \binom{m+s-1}{m}\right)$ *time algorithm that computes an encoding map of M given s, d, m, and $\mathbb{F}$.*

Lemma 3.7 ([KSY14, Lemma 3.6])**.** *Let M be the multiplicity code as above. Let $\delta = 1-\frac{d}{s\cdot|\mathbb{F}|}$ be a lower bound for the relative distance of M. Suppose $|\mathbb{F}| \geq \max\{10\cdot m, \frac{d+6\cdot s}{s}, 12\cdot(s+1)\}$. Then M is locally correctable from $\delta/10$ fraction of errors with query complexity $O(s^m\cdot|\mathbb{F}|)$.*

As discussed in Section 4.3 of [KSY14], this local corrector can be implemented to have running time $\operatorname{poly}(|\mathbb{F}|, s^m)$ over fields of constant characteristic. In fact, [Kop14] shows that the query complexity and running time for local correcting multiplicity codes can be further reduced to $|\mathbb{F}|\cdot O\left((\frac{1}{\delta})^m\right)$ queries, but this does not lead to any noticeable improvement for our setting.

We now prove Lemma 3.3.

Proof. Let $n \in \mathbb{N}$ be a codeword length. We set the code W_n to be a multiplicity code with the following parameters. We choose $\mathbb{F}$ to be a field of size $2^{\sqrt{\log n \cdot \log\log n}}$, and choose $m = \sqrt{\frac{\log n}{\log\log n}}$. Note that indeed $|\mathbb{F}|^m = n$. We choose $s = 2 \cdot m^2 \cdot \log n$. Let $\delta = \frac{1}{2 \cdot m \cdot \log n}$ (this will be a lower bound on the relative distance of the code) and choose the degree of the polynomials to be $d = s \cdot |\mathbb{F}| \cdot (1-\delta)$.

It can be verified that the relative distance of the code is at least $\delta \geq \Omega\left(\sqrt{\frac{\log\log n}{\log^3 n}}\right)$. The rate of the code is at least

$$\left(1 - \frac{m^2}{s}\right) \cdot (1-\delta)^m \geq \left(1 - \frac{1}{2 \cdot \log n}\right)\left(1 - \frac{1}{2 \cdot m \cdot \log n}\right)^m \geq 1 - \frac{1}{\log n},$$

as required. The alphabet size is

$$\begin{aligned} |\mathbb{F}|^{\binom{m+s-1}{m}} &\leq \exp\left(\sqrt{\log n \cdot \log\log n} \cdot s^m\right) \\ &= \exp\left(\sqrt{\log n \cdot \log\log n} \cdot \left(\frac{\log^2 n}{\log\log n}\right)^{\sqrt{\frac{\log n}{\log\log n}}}\right) \\ &= \exp\left(\exp\left(\sqrt{\log n \cdot \log\log n}\right)\right). \end{aligned}$$

Moreover, the alphabet is a vector space over $\mathbb{F}$ and hence in particular over $\mathbb{F}_2$ (since we chose the size of $\mathbb{F}$ to be a power of 2). The code W_n is $\mathbb{F}$-linear and in particular $\mathbb{F}_2$-linear.

By Lemma 3.7, W_n is locally correctable from $\frac{1}{10} \cdot \delta \geq \Omega\left(\sqrt{\frac{\log\log n}{\log^3 n}}\right)$ fraction of errors with query complexity

$$O(s^m \cdot |\mathbb{F}|) \leq O\left(\frac{\log^2 n}{\log\log n}\right)^{\sqrt{\frac{\log n}{\log\log n}}} \cdot 2^{\sqrt{\log n \cdot \log\log n}} = 2^{O(\sqrt{\log n \cdot \log\log n})},$$

as required. Finally, the fact that the family $\{W_n\}_n$ is explicit follows from the "furthermore" part of Lemma 3.6, and the fact that it has an efficient uniform local corrector with the required running time follows from the discussion after Lemma 3.7. ■

3.3 LDCs

As remarked earlier, by choosing a systematic encoding map, linear LCCs automatically give LDCs with the same rate, relative distance, and query complexity. The running time of the local decoding algorithm will be essentially the same as the running time of the local correction algorithm, provided that the systematic encoding map can be computed efficiently. Using the fact that multiplicity codes have an efficiently computable systematic encoding map [Kop12], it is easy to check that the codes we construct above also have an efficiently computable systematic encoding map. Thus we get LDCs with the same parameters as our LCCs.

4 LTCs with sub-polynomial query complexity

In this section, we prove the following theorem on LTCs, which immediately implies Theorem 1.4 from the introduction.

Theorem 4.1 (Main LTC theorem)**.** *For every $r \in (0,1)$, there exists an explicit infinite family of $\mathbb{F}_2$-linear codes $\{C_n\}_n$ satisfying:*

1. *C_n has block length n, rate at least r, and relative distance at least $1 - r - o(1)$.*
2. *C_n is locally testable with query complexity $\exp(\sqrt{\log n \cdot \log\log n})$.*
3. *The alphabet of C_n is a vector space Σ_n over $\mathbb{F}_2$, such that $|\Sigma_n| \le \exp(\exp\left(\sqrt{\log n \cdot \log\log n}\right))$.*

Furthermore, the family $\{C_n\}_n$ has a uniform local tester that runs in time $\exp(\sqrt{\log n \cdot \log\log n}))$.

We note that the existence of binary LTCs (Theorem 1.2) also follows from Theorem 4.1: In order to construct the binary LTCs, we concatenate the codes of Theorem 4.1 with any asymptotically good inner binary code that has efficient encoding and decoding algorithms. The local tester of the binary LTCs will emulate the original local tester, and whenever the latter queries a symbol, the binary local tester will emulate this query by reading the corresponding codeword of the inner code. If this string is not a legal codeword, the binary tester will reject, and otherwise it will decode the symbol and feed it to the original tester. Since such constructions are standard, we do not provide the full details.

The proof of Theorem 4.1 has two steps. In the first step, we give a transformation that amplifies the relative distance of an LTC – this step follows the distance amplification of [AL96]. In the second step, we construct a locally-testable code W_n with the desired query complexity but that has sub-constant relative distance. Finally, we construct the code C_n by applying the distance amplification to W_n. Those two steps are formalized in the following pair of lemmas, which are proved in Sections 4.2 and 4.3 respectively.

Lemma 4.2. *Suppose that there exists a code W with relative distance δ_W that is locally testable with query complexity q such that:*

- *W has rate r_W.*
- *W is $\mathbb{F}_2$-linear.*

Then, for every $0 < \delta, \varepsilon < 1$, there exists a code C with relative distance at least δ that is locally testable with query complexity $q \cdot \mathrm{poly}(1/(\varepsilon \cdot \delta_W))$, such that:

- *$|C| = |W|$.*
- *C has rate at least $r_W \cdot (1 - \delta - \varepsilon)$.*
- *Let Λ denote the alphabet of W. Then, the alphabet of C is $\Sigma \stackrel{\text{def}}{=} \Lambda^p$ for some $p = \mathrm{poly}(1/(\varepsilon \cdot \delta_W))$.*
- *C is $\mathbb{F}_2$-linear.*

Furthermore,

- *There is a polynomial time algorithm that computes a bijection from every code W to the corresponding code C, given r_W, δ_W, r, ε and Λ.*

- *There is an oracle algorithm that when given black box access to the local tester of any code W, and given also r_W, δ_W, r, ε, Λ, and the block length of W, computes the local tester of the corresponding code C. The resulting local tester of C runs in time that is polynomial in the running time of the local tester of W and in $1/\delta_W$, $1/\varepsilon$ and $\log(n_W)$ where n_W is the block length of W.*

Lemma 4.3. *There exists an explicit infinite family of $\mathbb{F}_2$-linear codes $\{W_n\}_n$ satisfying:*

1. *W_n has block length n, rate at least $1-\frac{1}{\log n}$, and relative distance at least $\exp(-\sqrt{\log n \cdot \log\log n})$.*
2. *W_n is locally testable with query complexity $\exp(\sqrt{\log n \cdot \log\log n})$.*
3. *The alphabet of W_n is a vector space Λ_n over $\mathbb{F}_2$, such that $|\Lambda_n| \leq \exp\left(\sqrt{\log n \cdot \log\log n}\right)$.*

Furthermore, the family $\{W_n\}_n$ has a uniform local tester that runs in time $\exp(\sqrt{\log n \cdot \log\log n})$.

Proof of Theorem 4.1. We construct the family $\{C_n\}_n$ by applying Lemma 4.2 to the family $\{W_n\}_n$ of Lemma 4.3 with $\delta_W = 2^{-O(\sqrt{\log n \cdot \log\log n})}$, $\varepsilon = \frac{1}{\log n}$ and

$$\delta = 1 - \frac{r}{1-\frac{1}{\log n}} - \varepsilon = 1 - r - O\left(\frac{1}{\log n}\right).$$

It is easy to see that C_n has the required rate, relative distance and alphabet size, and that it can be locally tested with the required query complexity. The family $\{C_n\}_n$ is explicit due to the first item in the "furthermore" part of Lemma 4.2, and has a uniform local corrector with the required running time due to the second item of that part. ■

Remark 4.4. In Lemma 4.2 above, as in Lemma 3.2, we chose to assume that W is $\mathbb{F}_2$-linear for simplicity. More generally, if W is $\mathbb{F}$-linear for any finite field $\mathbb{F}$, then C is $\mathbb{F}$-linear as well. Furthermore, the lemma also works if W is not $\mathbb{F}$-linear for any field $\mathbb{F}$, in which case C is not guaranteed to be $\mathbb{F}$-linear for any field $\mathbb{F}$.

4.1 Proof of Lemma 4.2

Our construction of the LTC C is the same as the construction of the LCCs of Section 3.1, with τ_W and τ replaced by $\delta_W/2$ and $\delta/2$ respectively. Our LTCs have the required rate, relative distance and alphabet size due to the same considerations as before[5].

It remains to prove that C is locally testable with query complexity $q \cdot \text{poly}(1/(\varepsilon \cdot \delta_W))$. To this end, we describe a local tester A. In what follows, we use the notation of Section 3.1.2.

Let A_W be the local tester of W. When given oracle access to a purported codeword $z \in \Sigma^n$, the local tester A emulates the action of A_W in the natural way: Recall that A_W expects to be given access to a purported codeword of W, and makes queries to it. Whenever A_W makes a query to a coordinate $j \in [n_W]$, the algorithm A performs the following steps:

1. A finds the block B_l to which the coordinate j belongs. Formally, $l \stackrel{\text{def}}{=} \lceil j/(b \cdot t) \rceil$.

[5]In particular, the lower bound on the relative distance of our LTC C follows from the lower bound on the relative distance given in Lemma 3.2, using the fact that our LTC W has a (trivial, inefficient) n_W query local corrector from $\delta_W/2$ fraction errors. Again, this lower bound on the distance could have been argued directly, without talking about locality.

2. A finds the neighbors of the vertex u_l in G_n. Let us denote those vertices by $v_{j_1}, \ldots, v_{j_d}$.

3. A queries the coordinates $j_1, \ldots j_d$, thus obtaining the blocks $S_{j_1}, \ldots, S_{j_d}$.

4. A reconstructs the block B_l by reversing the permutation of G_n on $S_{j_1}, \ldots, S_{j_d}$.

5. If B_l is not a codeword of $RS_{b,d}$, the local tester A rejects.

6. Otherwise, A retrieves the value of the j-th coordinate of w from B_l, and feeds it to A_W as an answer to its query.

If A_W finishes running, then A accepts if and only if A_W accepts.

It is easy to see that the query complexity of A is $d \cdot q$. It is also not hard to see that if z is a legal codeword of C, then A accepts with probability 1. It remains to show that if z is not a codeword of C then A rejects with probability at least $\operatorname{dist}(z, C)$. To this end, it suffices to prove that A rejects with probability at least $\frac{1}{\operatorname{poly}(d)} \cdot \operatorname{dist}(z, C)$ – as explained in Section 2.3, this rejection probability can be amplified to $\operatorname{dist}(z, C)$ while increasing the query complexity by a factor of $\operatorname{poly}(d)$, which is acceptable. We use the following definitions:

1. Let $S_1^z, \ldots, S_n^z \in \mathbb{F}^d$ be the blocks that correspond to the symbols of z.

2. Let $B_1^z, \ldots, B_n^z \in \mathbb{F}^d$ be the blocks that are obtained from $S_1^z, \ldots, S_n^z$ by reversing the permutation.

3. Let $w^z \in (\Lambda \cup \{?\})^{n_W}$ be the string that is obtained from the blocks $B_1^z, \ldots, B_n^z$ as follows: for each block B_l^z that is a legal codeword of $RS_{b,d}$, we extract from B_l^z the corresponding coordinates of w^z in the natural way. For each block B_l^z that is not a legal codeword of $RS_{b,d}$, we set the corresponding coordinates of w^z to be "?".

We would like to lower bound the probability that A rejects z in terms of the probability that A_W rejects w^z. However, there is a small technical problem: A_W is defined as acting on strings in Λ^{n_W}, and not on strings in $(\Lambda \cup \{?\})^{n_W}$. To deal with this technicality, we define an algorithm A'_W that, when given access to a string $y \in (\Lambda \cup \{?\})^{n_W}$, emulates A_W on y, but rejects whenever a query is answered with "?". We use the following proposition, whose proof we defer to Section 4.1.1.

Proposition 4.5. *A'_W rejects a string $y \in (\Lambda \cup \{?\})^{n_W}$ with probability at least*

$$\frac{1}{2} \cdot \min \{\operatorname{dist}(y, W), \delta_W\}.$$

Now, it is not hard to see that when A is invoked on z, it emulates the action of A'_W on w^z. To finish the proof, note that since each coordinate in W has at most d coordinates of C that depend on it, it holds that

$$\operatorname{dist}(z, C) \cdot n \leq d \cdot \operatorname{dist}(w^z, W) \cdot n_W$$

and therefore

$$\operatorname{dist}(w^z, W) \quad \geq \quad \frac{n}{n_W} \cdot \frac{1}{d} \cdot \operatorname{dist}(z, C) \geq \frac{1}{b \cdot t \cdot d} \cdot \operatorname{dist}(z, C).$$

It thus follows that A rejects z with probability at least

$$\frac{1}{2} \cdot \min \{\operatorname{dist}(w^z, W), \delta_W\} \geq \frac{1}{\operatorname{poly}(d)} \cdot \operatorname{dist}(z, C),$$

as required.

It is not hard to see that the local tester A can be implemented efficiently with black box access to A_W, as required by the second item in the "furthermore" part of the lemma.

4.1.1 Proof of Proposition 4.5

We use the following result.

Claim 4.6. *Let $I \subseteq [n_W]$ be a set of coordinates. The algorithm A_W queries some coordinate in I with probability at least*

$$\min \left\{ \frac{|I|}{n_W}, \frac{1}{2} \cdot \delta_W \right\}.$$

Note that this claim only makes sense since we assumed that A_W makes *non-adaptive* queries (we assumed it in Definition 2.5). Without this assumption, the probability that A_W queries some coordinate in I would have depended on the tested string.

Proof. It suffices to prove that for every $I \subseteq [n_W]$ such that $\frac{|I|}{n_W} \le \frac{1}{2} \cdot \delta_W$, the algorithm A_W queries some coordinate in I with probability at least $\frac{|I|}{n_W}$. Let I be such a set, and let $s \in \Lambda^{n_W}$ be an arbitrary string that contains non-zero values inside I, and contains 0 everywhere outside I. Clearly,

$$\mathrm{dist}(s, W) = \frac{|I|}{n_W},$$

and therefore A_W rejects s with probability at least $\frac{|I|}{n_W}$. On the other hand, A_W can only reject s if it queries some coordinate in I, since otherwise it cannot distinguish between s and the all-zeroes codeword. It follows that A_W queries some coordinate in I with probability at least $\frac{|I|}{n_W}$, as required. ■

We turn to proving Proposition 4.5. Let

$$E \stackrel{\text{def}}{=} \{i : y_i = ?\}$$

be the set of erasures in y. We consider two cases:

- *E* **is "large":** Suppose that $\frac{|E|}{n_W} \ge \frac{1}{2} \cdot \mathrm{dist}(y, W)$. In this case, it holds by Claim 4.6 that A_W queries some coordinate in E with probability at least

$$\frac{1}{2} \cdot \min \{\mathrm{dist}(y, W), \delta_W\}.$$

  Since A'_W rejects y whenever A_W queries some coordinate in E, the proposition follows.

- *E* **is "small":** Suppose that $\frac{|E|}{n_W} \le \frac{1}{2} \cdot \mathrm{dist}(y, W)$. Let $y_0 \in \Lambda^{n_W}$ be an arbitrary string that agrees with y outside E. Clearly,

$$\mathrm{dist}(y, W) \le \mathrm{dist}(y_0, W) + \frac{|E|}{n_W},$$

so $\mathrm{dist}(y_0, W) \geq \frac{1}{2} \cdot \mathrm{dist}(y, W)$. Let $\mathcal{E}$ denote the event that A_W queries some coordinate in E. We have that

$$\begin{aligned}
\Pr\left[A'_W \text{ rejects } y\right] &= \Pr\left[\mathcal{E}\right] \cdot \Pr\left[A'_W \text{ rejects } y | \mathcal{E}\right] + \Pr\left[\neg\mathcal{E}\right] \cdot \Pr\left[A'_W \text{ rejects } y | \neg\mathcal{E}\right] \\
&= \Pr\left[\mathcal{E}\right] \cdot 1 + \Pr\left[\neg\mathcal{E}\right] \cdot \Pr\left[A_W \text{ rejects } y_0 | \neg\mathcal{E}\right] \\
&\geq \Pr\left[\mathcal{E}\right] \cdot \Pr\left[A_W \text{ rejects } y_0 | \mathcal{E}\right] + \Pr\left[\neg\mathcal{E}\right] \cdot \Pr\left[A_W \text{ rejects } y_0 | \neg\mathcal{E}\right] \\
&= \Pr\left[A_W \text{ rejects } y_0\right] \\
&\geq \mathrm{dist}(y_0, W) \\
&\geq \frac{1}{2} \cdot \mathrm{dist}(y, W),
\end{aligned}$$

as required.

This concludes the proof.

4.2 Proof of Lemma 4.3

In this section, we prove Lemma 4.3, restated below.

Lemma 4.3. *There exists an explicit infinite family of $\mathbb{F}_2$-linear codes $\{W_n\}_n$ satisfying:*

1. *W_n has block length n, rate at least $1 - \frac{1}{\log n}$, and relative distance at least $\exp(-\sqrt{\log n \cdot \log\log n})$.*
2. *W_n is locally testable with query complexity $\exp(\sqrt{\log n \cdot \log\log n})$.*
3. *The alphabet of W_n is a vector space Λ_n over $\mathbb{F}_2$, such that $|\Lambda_n| \leq \exp\left(\sqrt{\log n \cdot \log\log n}\right)$.*

Furthermore, the family $\{W_n\}_n$ has a uniform local tester that runs in time $\exp(\sqrt{\log n \cdot \log\log n})$.

For the proof of Lemma 4.3 we use the *tensor product codes* instantiated in the sub-constant relative distance regime. The use of tensor products to construct LTCs was initiated by [BS06], and was studied further in [Val05, DSW06, BV09b, BV09a, Vid11]. Our construction is based on a result of [Vid11].

We start with some definitions. Let $\mathbb{F}$ be a finite field. For a pair of vectors $h_1 \in \mathbb{F}^{\ell_1}$ and $h_2 \in \mathbb{F}^{\ell_2}$ their *tensor product* $h_1 \otimes h_2$ denotes the matrix $M \in \mathbb{F}^{\ell_1 \times \ell_2}$ with entries $M_{(i_1,i_2)} = (h_1)_{i_1} \cdot (h_2)_{i_2}$ for every $i_1 \in [\ell_1]$ and $i_2 \in [\ell_2]$. For a pair of linear codes $H_1 \subseteq \mathbb{F}^{\ell_1}$ and $H_2 \subseteq \mathbb{F}^{\ell_2}$ their *tensor product code* $H_1 \otimes H_2 \subseteq \mathbb{F}^{\ell_1 \times \ell_2}$ is defined to be the linear subspace spanned by all matrices of the form $h_1 \otimes h_2$ where $h_1 \in H_1$ and $h_2 \in H_2$. For a linear code H, let $H^1 = H$ and $H^m = H^{m-1} \otimes H$. The following are some useful facts regarding tensor product codes (see e.g. [DSW06]).

Fact 4.7. *Let $H_1 \subseteq \mathbb{F}^{\ell_1}$ and $H_2 \subseteq \mathbb{F}^{\ell_2}$ be linear codes of rates r_1, r_2 and relative distances δ_1, δ_2 respectively. Then $H_1 \otimes H_2 \subseteq \mathbb{F}^{\ell_1 \times \ell_2}$ is a linear code of rate $r_1 \cdot r_2$ and relative distance $\delta_1 \cdot \delta_2$. In particular, if $H \subseteq \mathbb{F}^{\ell}$ is a linear code of rate r and relative distance δ then $H^m \subseteq \mathbb{F}^{\ell^m}$ is a linear code of rate r^m and relative distance δ^m.*

We use the following theorem that is given as Corollary 3.6 in [Vid11].

Theorem 4.8 (Immediate corollary of [Vid11, Thm. 3.1])**.** *Let $H \subseteq \mathbb{F}^{\ell}$ be a linear code with relative distance δ. Then for every $m \geq 3$, the code $H^m \subseteq \mathbb{F}^{\ell^m}$ is locally testable with query complexity*

$$\ell^2 \cdot \mathrm{poly}(m)/\delta^{2m}.$$

For the proof of Lemma 4.3, we instantiate Theorem 4.8 with the tensor product of Reed-Solomon[6] codes.

Proof of Lemma 4.3 Fix a codeword length $n \in \mathbb{N}$. The code W_n is defined as follows. Let $\mathbb{F} \stackrel{\text{def}}{=} \mathbb{F}_{2^{\sqrt{\log n \cdot \log\log n}}}$, and let $m \stackrel{\text{def}}{=} \sqrt{\frac{\log n}{\log\log n}}$. Let R be a Reed-Solomon code over $\mathbb{F}$ with block length $n^{1/m}$, rate $r \stackrel{\text{def}}{=} \left(1 - \frac{1}{\log n}\right)^{1/m}$ and relative distance $1-r$. Note that indeed the block length is at most $|\mathbb{F}|$, which is required for the existence of such codes. Finally, let $W_n = R^m$.

From the properties of tensor codes we have that W_n is a linear code over $\mathbb{F}$ with block length $(n^{1/m})^m = n$, rate $r^m = 1 - \frac{1}{\log n}$, and relative distance

$$
\begin{aligned}
\left(1-r\right)^m &= \left(1 - \left(1 - \frac{1}{\log n}\right)^{1/m}\right)^m \\
&\geq \left(1 - \left(1 - \frac{1}{4 \cdot m \cdot \log n}\right)\right)^m \qquad \left(\text{Fact 2.1}: (1-x)^y \leq 1 - \tfrac{1}{4} \cdot x \cdot y\right) \\
&= \left(\frac{1}{4 \cdot m \cdot \log n}\right)^m \\
&= 2^{-O(m \cdot (\log m + \log\log n))} \\
&= 2^{-O(\sqrt{\log n \cdot \log\log n})},
\end{aligned}
$$

as required. The fact that W_n can be encoded in time $\mathrm{poly}(n)$ follows from standard properties of tensor product codes (see e.g. [Sud01, Lecture 6]).

Finally, by Theorem 4.8, we have that W_n is locally testable with query complexity at most

$$
n^{2/m} \cdot \mathrm{poly}(m) \cdot \left(\frac{1}{4 \cdot m \cdot \log n}\right)^{-2m} = 2^{O(\sqrt{\log n \cdot \log\log n})},
$$

as required. The fact that the family $\{W_n\}_n$ has a uniform local tester with the required running time follows immediately from the proof of [Vid11]. ■

5 Open Questions

We conclude with some open questions.

- In this work we found that LCCs and LTCs with sub-constant relative distance can be useful. Are there better LCCs and LTCs in the sub-constant relative distance regime?
- LCCs and LTCs often come together with PCPs. Can we construct constant-rate PCPs with sub-polynomial query complexity?
- Are there applications of our LCCs and LTCs to complexity theory?

Acknowledgement. We would like to thank Irit Dinur, Tali Kaufman, Ran Raz and Avi Wigderson for useful discussions and ideas. We would also like to thank Oded Goldreich, Irit Dinur, Madhu Sudan and anonymous referees for helpful comments on the preliminary version of this work.

[6] We chose Reed-Solomon codes for convenience, but any high-rate codes with reasonable distance will do.

References

[AC88] Noga Alon and Fan R. K. Chung. Explicit construction of linear sized tolerant networks. *Discrete Mathematics*, 72(1-3):15–19, 1988.

[AL96] Noga Alon and Michael Luby. A linear time erasure-resilient code with nearly optimal recovery. *IEEE Transactions on Information Theory*, 42(6):1732–1736, 1996.

[ALM+98] Sanjeev Arora, Carsten Lund, Rajeev Motwani, Madhu Sudan, and Mario Szegedy. Proof verification and intractability of approximation problems. *Journal of ACM*, 45(3):501–555, 1998. Preliminary version in FOCS 1992.

[AM85] N. Alon and V. D. Milman. λ_1, lsoperimetric inequalities for graphs, and superconcentrators. *JOURNAL OF COMBINATORIAL THEORY, Series B*, 38(1):73–88, 1985.

[AS98] Sanjeev Arora and Shmuel Safra. Probabilistic checkable proofs: A new characterization of NP. *Journal of ACM volume*, 45(1):70–122, 1998. Preliminary version in FOCS 1992.

[BFLS91] László Babai, Lance Fortnow, Leonid A. Levin, and Mario Szegedy. Checking computations in polylogarithmic time. In *STOC*, pages 21–31, 1991.

[BS06] Eli Ben-Sasson and Madhu Sudan. Robust locally testable codes and products of codes. *Random Struct. Algorithms*, 28(4):387–402, 2006. Preliminary version in APPROX-RANDOM 2004.

[BS08] Eli Ben-Sasson and Madhu Sudan. Short PCPs with polylog query complexity. *SIAM J. Comput.*, 38(2):551–607, 2008. Preliminary version in STOC 2005.

[BSV12] Eli Ben-Sasson and Michael Viderman. Towards lower bounds on locally testable codes via density arguments. *Computational Complexity*, 21(2):267–309, 2012.

[BV09a] Eli Ben-Sasson and Michael Viderman. Composition of semi-LTCs by two-wise tensor products. In *APPROX-RANDOM*, pages 378–391, 2009.

[BV09b] Eli Ben-Sasson and Michael Viderman. Tensor products of weakly smooth codes are robust. *Theory of Computing*, 5(1):239–255, 2009.

[Din07] Irit Dinur. The PCP theorem by gap amplification. *Journal of ACM*, 54(3):241–250, 2007. Preliminary version in STOC 2006.

[DK11] Irit Dinur and Tali Kaufman. Dense locally testable codes cannot have constant rate and distance. In *APPROX-RANDOM*, pages 507–518, 2011.

[Dod84] Jozef Dodziuk. Difference equations, isoperimetric inequality and transience of certain random walks. *Transactions of the American Mathematical Society*, 284(2):787–794, 1984.

[DSW06] Irit Dinur, Madhu Sudan, and Avi Wigderson. Robust local testability of tensor products of ldpc codes. In *APPROX-RANDOM*, pages 304–315, 2006.

[Efr12] Klim Efremenko. 3-query locally decodable codes of subexponential length. *SIAM J. Comput.*, 41(6):1694–1703, 2012.

[FS95] Katalin Friedl and Madhu Sudan. Some improvements to total degree tests. In *ISTCS*, pages 190–198, 1995.

[GG81] Ofer Gabber and Zvi Galil. Explicit constructions of linear-sized superconcentrators. *J. Comput. Syst. Sci.*, 22(3):407–420, 1981.

[GI02] Venkatesan Guruswami and Piotr Indyk. Near-optimal linear-time codes for unique decoding and new list-decodable codes over smaller alphabets. In *STOC*, pages 812–821, 2002.

[GI05] Venkatesan Guruswami and Piotr Indyk. Linear-time encodable/decodable codes with near-optimal rate. *IEEE Transactions on Information Theory*, 51(10):3393–3400, 2005.

[Gil52] Edgar N. Gilbert. A comparision of signalling alphabets. *Bell System Technical Journal*, 31:504–522, 1952.

[GKS13] Alan Guo, Swastik Kopparty, and Madhu Sudan. New affine-invariant codes from lifting. In *ITCS*, pages 529–540, 2013.

[GR08] Venkatesan Guruswami and Atri Rudra. Explicit codes achieving list decoding capacity: Error-correction with optimal redundancy. *IEEE Transactions on Information Theory*, 54(1):135–150, 2008.

[GS00] Oded Goldreich and Shmuel Safra. A combinatorial consistency lemma with application to proving the PCP theorem. *SIAM J. Comput.*, 29(4):1132–1154, 2000.

[GS06] Oded Goldreich and Madhu Sudan. Locally testable codes and PCPs of almost linear length. *Journal of ACM*, 53(4):558–655, 2006. Preliminary version in FOCS 2002, pages 13-22.

[HLW06] Shlomo Hoory, Nati Linial, and Avi Wigderson. Expander graphs and their applications. *Bulletin of AMS*, 43(4):439–561, 2006.

[HOW13] Brett Hemenway, Rafail Ostrovsky, and Mary Wootters. Local correctability of expander codes. In *ICALP (1)*, pages 540–551, 2013.

[Kop12] S. Kopparty. List-decoding multiplicity codes. In *Electronic Colloquium on Computational Complexity (ECCC)*, TR12-044, 2012.

[Kop14] Swastik Kopparty. Some remarks on multiplicity codes. In *Proceedings of the AMS Special Session on Discrete Geometry and Algebraic Combinatorics*, Contemporary Mathematics, 2014.

[KSY14] Swastik Kopparty, Shubhangi Saraf, and Sergey Yekhanin. High-rate codes with sublinear-time decoding. *J. ACM*, 61(5):28, 2014.

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

[KY09] Kiran S. Kedlaya and Sergey Yekhanin. Locally decodable codes from nice subsets of finite fields and prime factors of mersenne numbers. *SIAM J. Comput.*, 38(5):1952–1969, 2009.

[Mei14] Or Meir. Locally correctable and testable codes approaching the singleton bound. *Electronic Colloquium on Computational Complexity (ECCC)*, 21:107, 2014.

[Rag07] Prasad Raghavendra. A note on yekhanin's locally decodable codes. *Electronic Colloquium on Computational Complexity (ECCC)*, 14(016), 2007.

[RS60] Irving S. Reed and Gustave Solomon. Polynomial codes over certain finite fields. *SIAM Journal of the Society for Industrial and Applied Mathematics*, 8(2):300–304, 1960.

[RS96] Ronitt Rubinfeld and Madhu Sudan. Robust characterization of polynomials with applications to program testing. *SIAM Journal of Computing*, 25(2):252–271, 1996.

[STV01] Madhu Sudan, Luca Trevisan, and Salil P. Vadhan. Pseudorandom generators without the xor lemma. *J. Comput. Syst. Sci.*, 62(2):236–266, 2001.

[Sud01] Madhu Sudan. Algorithmic introduction to coding theory (lecture notes), 2001.

[Val05] Paul Valiant. The tensor product of two codes is not necessarily robustly testable. In *APPROX-RANDOM*, pages 472–481, 2005.

[Var57] R. R. Varshamov. Estimate of the number of signals in error correcting codes. *Doklady Akadamii Nauk*, pages 739–741, 1957.

[Vid11] Michael Viderman. A combination of testability and decodability by tensor products. *Electronic Colloquium on Computational Complexity (ECCC)*, 18:87, 2011.

[Vid15] Michael Viderman. Explicit strong LTCs with inverse poly-log rate and constant soundness. *Electronic Colloquium on Computational Complexity (ECCC)*, 20, 2015.

[WdW05] Stephanie Wehner and Ronald de Wolf. Improved lower bounds for locally decodable codes and private information retrieval. In *Automata, Languages and Programming, 32nd International Colloquium, ICALP 2005, Lisbon, Portugal, July 11-15, 2005, Proceedings*, pages 1424–1436, 2005.

[Woo07] David P. Woodruff. New lower bounds for general locally decodable codes. *Electronic Colloquium on Computational Complexity (ECCC)*, 14(006), 2007.

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

[Yek12] Sergey Yekhanin. Locally decodable codes. *Foundations and Trends in Theoretical Computer Science*, 6(3):139–255, 2012.

ECCC ISSN 1433-8092

http://eccc.hpi-web.de