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

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

January 10, 2017

Abstract

Locally-correctable codes (LCCs) and locally-testable codes (LTCs) are error-correcting codes that admit *local* algorithms for correction and detection of errors. Those algorithms are local in the sense that they only query a small number of entries of the corrupted codeword. The fundamental question about LCCs and LTCs is to determine the optimal tradeoff between their rate, distance and query complexity.

In this work, we construct the first LCCs and LTCs with constant rate, constant relative distance, and sub-polynomial query complexity. Specifically, we show that there exist LCCs and LTCs with block length n, constant rate (which can even be taken arbitrarily close to 1) and constant relative distance, whose query complexity is $\exp(\tilde{O}(\sqrt{\log n}))$ (for LCCs) and $(\log n)^{O(\log\log n)}$ (for LTCs).

In addition to having small query complexity, our codes also achieve better trade-offs between the rate and the relative distance than were previously known to be achievable by LCCs or LTCs. Specifically, over large (but constant size) alphabet, our codes approach the Singleton bound, that is, they have almost the best-possible relationship between their rate and distance. Over the binary alphabet, our codes meet the Zyablov bound. Such trade-offs between the rate and the relative distance were previously not known for any $o(n)$ query complexity. Our results on LCCs also immediately give locally-decodable codes (LDCs) with the same parameters.

*This paper is a merge of reports [KMRS15a] and [KMRS15b]. An extended abstract has appeared in proceedings of the 48th Annual ACM Symposium on Theory of Computing (STOC 2016), and a preliminary version appeared as report [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, Haifa University, Haifa 31905, Israel. ormeir@cs.haifa.ac.il. This research was carried out when Meir was supported by Irit Dinur's ERC grant number 239986 and in part by the Israel Science Foundation (grant No. 460/05).

§School of Mathematics, Institute for Advanced Study, Princeton, NJ, USA and DIMACS, Rutgers University, Piscataway, NJ, USA. This research was partially supported by NSF grants CCF-1412958 and CCF-1445755 and the Rothschild fellowship. 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

1 Introduction

An error-correcting code is a scheme for robust representation of data that enables detection and correction of data corruptions. More specifically, an error-correcting code C maps a message x of length k to a longer codeword $c \in C$ of length n that allows one to do the following: (1) Determine whether a given word z is corrupted (that is, whether it is an image of C), and (2) Recover the original codeword c (as well as the original message x) from the given word z in the case the latter is corrupted. There are two fundamental parameters of an error-correcting code:

- The rate is defined as the ratio k/n, and measures the amount of redundancy in communication.
- The relative distance is the minimum fraction of coordinates on which any two codewords differ. It is related to the error detection and correction capabilities of the code.

A remarkable fact [Sha48, Ham50] is that there exist good error-correcting codes with constant rate and *constant relative distance* where the constants are independent of the codeword length n.

The traditional error-detection and error-correction algorithms require one to read all the coordinates of the corrupted codeword. Since the groundbreaking work of [Sha48, Ham50] there has been remarkable progress in the algorithmic front providing explicit constructions of good error-correcting codes that can detect and correct errors efficiently in polynomial time, and even in linear time. In recent years, new families of codes were discovered that can detect and correct errors, with high probability, in *sublinear time* by examining only a small number of coordinates of the corrupted codeword. 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 algorithms is called the query complexity. A major objective of study with regard to LCCs and LTCs is to understand the fundamental trade-off between their rate, distance, and query complexity.

LCCs and LTCs were originally studied in the setting where the query complexity was constant (or poly-logarithmic) and the relative distance was constant. Besides being interesting in its own right, this regime was motivated by many applications of such codes in complexity theory, such as program checking [BK95, Lip90, BLR93, RS96], probabilistically checkable proofs [BFLS91, AS98, ALM+98, GS06], derandomization, hardness amplification and worst-case to average-case reductions [BFNW93, STV01, Tre03], and private information retrieval [CKGS98]..

In the constant query regime, it is believed that LCCs and LTCs must be very redundant, since every bit of the codeword must contain, in some sense, information about many other bits of the codeword. Hence, we do not expect such codes to achieve a high rate. In particular, in this setting it is known that linear LCCs cannot have constant rate [KT00, WW05, Woo07], and that LTCs with

[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.

certain restrictions cannot have constant rate [DK11, BV12]. On the other hand, the best-known constant-query LCCs have exponential length (for example, a constant-degree Reed-Muller code is such an LCC), and the best-known constant-query LTCs have quasi-linear length [BS08, Din07].

This paper is about LCCs and LTCs in the standard coding-theoretic regime of constant rate and constant relative distance with sublinear query complexity (that may be super-constant). This regime is of potential interest in practical applications of coding theory to data storage and transmission, as using codes of small rate translates into increased storage space or transmission time, and is infeasible in most such applications. On the other hand, for such applications it is typically not crucial that the query complexity is constant.

In the constant rate regime, it has long been known that Reed-Muller codes are LCCs and LTCs with query complexity $O(n^\beta)$ for any constant $\beta > 0$ [BFLS91, RS96]. More recently, it has been discovered that both LCCs and LTCs can even achieve rates that are arbitrarily close to 1 with query complexity $O(n^\beta)$ for an arbitrary constant $\beta > 0$. This was achieved using quite a few different constructions such as multiplicity codes [KSY14], affine-invariant codes [GKS13] and expander codes [HOW15] (for LCCs), and such as tensor codes [BS06, BV15, Vid15] and affine-invariant codes [GKS13] (for LTCs). These results were surprising since they were 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, as mentioned above, there were several quite different constructions of LCCs and LTCs with constant rate and constant relative distance, all of which had $\Omega(n^\beta)$ query complexity. Specifically:

- For LCCs, we obtain query complexity and running time $\exp(\sqrt{\log n \cdot \log\log n})$.
- For LTCs, we obtain query complexity and running time $(\log n)^{O(\log\log n)}$.

Furthermore, we show that such codes can achieve stronger trade-offs between the rate and relative distance than were known before. Specifically, over large alphabets (of constant size), our codes approach the Singleton bound: they achieve a tradeoff between rate and distance which is essentially as good as possible for general error-correcting codes. 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. Over the binary alphabet, our codes meet the Zyablov bound. Such trade-offs were previously not known for any $o(n)$ query complexity.

1.1 Main results

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 exists $Z(r) \in (0,1)$ such that there exists 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 $Z(r)$.*
2. *C_n is locally correctable from $\frac{1}{2} \cdot Z(r)$ 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 exists $Z(r) \in (0,1)$ such that there exists 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 $Z(r)$.*
2. *C_n is locally testable with query complexity and running time at most $(\log n)^{O(\log\log n)}$.*

Our proofs in fact show that $Z(r)$ can be taken to equal any real number smaller than

$$\max_{R\in(r,1)} \left\{ (1-R) \cdot H^{-1}(1-\frac{r}{R}) \right\},$$

where $H(x) = x\log x + (1-x)\log(1-x)$ is the binary entropy function and H^{-1} is the inverse of H in the domain $(0, \frac{1}{2})$. Thus the codes in the above theorems match the Zyablov bound..
The above codes over the binary alphabet are obtained by first constructing LCCs and LTCs over large alphabets that approach the *Singleton bound* [Sin64], and then concatenating them with binary codes that match the *Gilbert-Varshamov bound* [Gil52, Var57]. The following theorems describe these large-alphabet LCCs and LTCs.

Theorem 1.3 (LCCs with sub-polynomial query complexity approaching the Singleton bound)**.** *For every $r \in (0,1)$ and $\varepsilon > 0$, there exist a finite alphabet $\Sigma = \mathbb{F}_2^s$ and an explicit infinite family of $\mathbb{F}_2$-linear codes $\{C_n\}_n$ over Σ satisfying:*

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

Theorem 1.4 (LTCs with sub-polynomial query complexity approaching the Singleton bound)**.** *For every $r \in (0,1)$ and $\varepsilon > 0$, there exists a finite alphabet $\Sigma = \mathbb{F}_2^s$ and an explicit infinite family of $\mathbb{F}_2$-linear codes $\{C_n\}_n$ over Σ satisfying:*

1. *C_n has block length n, rate at least r, and relative distance at least $1-r-\varepsilon$,*
2. *C_n is locally testable with query complexity and running time at most $(\log n)^{O(\log\log n)}$,*
3. *The size of the alphabetΣ is at most $\exp(\mathrm{poly}(1/\varepsilon))$.*

The above theorems are proved in Sections 3, 4 and 5. Note that the exponential dependence of the alphabet size on ε follows from our use of the distance-amplification method of Alon, Edmonds, and Luby (see below). This dependence seems to be a bottleneck in all applications of this method, e.g. [GI05].

1.2 Our techniques

The proofs of Theorems 1.3 and 1.4 follow the same basic outline:

- First, we construct codes with very poor relative distance and very high rate that have the desired query complexity. Specifically, the relative distance will be *sub-constant*.
- Then, we amplify the relative distance of the latter codes, thus obtaining new codes that approach the Singleton bound. This is done by employing a construction of Alon, Edmonds and Luby [AEL95, AL96].

Our main contributions in those proofs could be summarized as follows:

- We observe that the construction of [AEL95, AL96] could be viewed as a distance-amplification technique.
  In particular, we observe that this technique can be used to amplify the relative distance of LCCs and LTCs without increasing their query complexity by too much.
- We observe that this technique works even for codes with *sub-constant* relative distance.
- We show that it is possible to construct LCCs and LTCs with better query complexity by allowing the relative distance to be sub-constant.

We now elaborate on each of those points.

1.2.1 Distance amplification

As mentioned above, our distance amplification method is based on a construction of Alon, Edmonds, and Luby [AEL95, AL96]. We observe that this technique can be viewed as a method for distance amplification. 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 [AEL95, AL96] was for constructing linear-time erasure-decodable codes approaching the Singleton bound. In addition to the above distance-amplification technique, [AEL95, AL96] constructed a linear-time erasure-decodable code (not approaching the Singleton bound but with constant relative distance) which could be used as the input code to the amplifier. The main result of [AEL95, 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 achieve the Singleton bound, and in [GI02, GR08], who constructed capacity-achieving list-decodable codes with

constant-size alphabet. For the sake of brevity, throughout the rest of this paper, we refer to this technique as the "AEL distance-amplification".

Our first main observation is that the distance-amplification procedure 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 $\Theta(1/d)$. 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. As far as we are aware, there have been no previous uses of the AEL distance-amplification technique using an expander of super-constant degree.

The generality of the AEL technique. We wish to draw attention to the AEL technique. It can 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. This is probably the main "take-home message" from this work. Recently, following a preliminary version of this work [Mei14], Hemenway and Wootters [HW15] used this observation on the generality of the AEL technique to construct linear-time list-recoverable codes.

1.2.2 LCCs and LTCs with sub-polynomial query complexity and sub-constant relative distance

To obtain our final results, we show that for suitably constructed families of LCCs and LTCs one can achieve sub-polynomial query complexity by allowing sub-constant relative distance. We note that Reed-Muller codes, which up to very recently were essentially the only known LCCs and LTCs with constant rate and constant relative distance, cannot achieve sub-polynomial query complexity, even when allowing the relative distance to be sub-constant. However, we observe that this is possible using more recent constructions such as multiplicity and tensor codes.

In order to obtain our LCCs, we show that an existing family of high-rate LCCs can achieve sub-polynomial query complexity if we only allow them to have sub-constant relative distance. Specifically, multiplicity codes [KSY14] in a super-constant number of variables give us the desired LCCs.

In order to obtain our LTCs, we combine the AEL distance amplification and tensor products of codes. This construction builds on a long line of research on using tensor products of codes for the construction of LTCs, which was was initiated in [BS06]. We discuss this construction in detail in the following section.

1.2.3 Our construction of LTCs

We construct our improved LTCs in the sub-constant relative distance regime using an iterative strategy, along the lines of the construction of Meir [Mei09] (which itself is in the spirit of several

classical, iterative, "brave, yet moderate" algorithms [Gol11]: the zig-zag product [RVW02], undirected connectivity in log-space [Rei08] and the PCP theorem via gap amplification [Din07]). The main new ingredient in our iterative strategy is again the AEL distance-amplification technique (but in a different setting of parameters).

The iterative construction. Following [Mei09], our construction starts with a code of very small block length, which can be tested simply by reading the entire received word. Then, the block length is increased iteratively, while the rate, relative distance, and query complexity are not harmed by too much.

More specifically, suppose we want to construct a code with block length n. We start with a code of block length $\operatorname{poly}\log n$, rate $1 - \frac{1}{\operatorname{poly}\log n}$, and relative distance $\frac{1}{\operatorname{poly}\log n}$. Then, in each iteration, the block length and rate are (roughly) squared, the relative distance is maintained, and the query complexity is increased by a factor of $\operatorname{poly}\log n$. Thus, after $\approx \log\log n$ iterations, we obtain an LTC with block length n, constant rate, relative distance $\frac{1}{\operatorname{poly}\log n}$, and query complexity $(\log n)^{O(\log\log n)}$, as required.

A single iteration. We now describe the structure of a single iteration. Suppose that, at the beginning of the iteration, the code C has block length n, rate r, relative distance δ, and query complexity q. We apply to C the following two operations:

- **Tensor product:** We replace C with its tensor product C^2. The tensor product C^2 is the code that contains all $n \times n$ matrices M such that that all rows of M and all columns of M are codewords of C. The code C^2 has block length n^2, rate r^2, relative distance δ^2, and query complexity $q \cdot \operatorname{poly}(1/\delta)$.
- **Distance amplification:** We apply (a variant of) the AEL distance-amplification to the code C^2, and amplify the relative distance from δ^2 to δ. The resulting code has block length $O(n^2)$, rate $(1-\delta) \cdot r^2$, relative distance δ, and query complexity $q \cdot \operatorname{poly}(1/\delta)$.

The iteration ends after the distance amplification.

The local testability of the tensor product. There is one more complication that needs to be handled: as explained above, we need the tensor product to preserve the local testability, i.e., to increase the query complexity by a factor of at most $\operatorname{poly}(1/\delta)$. However, this does not necessarily hold if C is an arbitrary code. In fact, there is a long line of research that attempts to understand the conditions under which tensor product preserves the local testability [BS06, Val05, CR05, DSW06, GM12, BV09, BV15, Vid15].

In order to resolve this issue, we use the following idea of [Mei09]. We say that a code C_0 has *property* $\square$ if there exists a code D such that C_0 is the tensor product D^2. It follows from [BS06, Vid15] that the tensor product operation roughly preserves the local testability of codes that have property $\square$. Specifically, if C_0 has property $\square$ and is locally testable with query complexity q and has relative distance δ, then C_0^2 is locally testable with query complexity $q \cdot \operatorname{poly}(1/\delta)$.

Hence, in order to make our construction go through, we maintain the invariant that our code C has property $\square$ throughout the iterations. This requires us to show that a single iteration preserves the property $\square$ of the code. To this end, first observe that the tensor product operation clearly preserves the property $\square$. The more challenging part is to make sure that the distance amplification

preserves the property $\square$. In order to do so, we define a new operation, which we called $\square$-distance amplification, which amplifies the distance while preserving the property $\square$ and the local testability.

The $\square$-distance amplification. We conclude by describing how the $\square$-distance amplification works: Suppose that we are given a code C_0 that has the property $\square$, and we wish to amplify its relative distance to δ. By definition, there exists some code D such that $C_0 = D^2$. We apply the AEL distance-amplification to D to obtain a new code D' with relative distance $\sqrt{\delta}$. We now define the code $C_0' = (D')^2$ to be the result of applying the $\square$-distance amplification to C_0. Observe that C_0' indeed has relative distance δ, and that it has the property $\square$, as required.

It remains to prove that the $\square$-distance amplification preserves the local testability, i.e., that this operation increases the query complexity by a factor of at most $\text{poly}(1/\delta)$. Following [Mei09], we show it by decomposing this operation into simpler block-wise operations, and showing that each of these operations preserves local testability.

Comparison with [Mei09]. The construction of [Mei09] yields LTCs with *constant* relative distance, *sub-constant* rate, and *constant* query complexity. This construction, too, starts with a code of small block length, which is trivially locally testable, and increases it iteratively. A single iteration of [Mei09] consists, too, of applying tensor product and distance amplification, as well as another operation called "random projection".

The result of [Mei09] is incomparable to ours: in particlar, our construction has a better rate, while the construction of [Mei09] has a better query complexity. The reason that the construction of [Mei09] has worse rate is that it loses a constant factor in the rate in each iteration. This loss is due to two bottlenecks:

- First, [Mei09] maintains a constant relative-distance throughout the iterations. When working in the regime of constant relative-distance, both the tensor product and the distance amplification decrease the rate by (at least) a constant factor. We, on the other hand, observe that by allowing the codes to have sub-constant relative distance, both operations can be made to lose only a $1 - o(1)$ factor in the rate at each iteration.
  One reason that [Mei09] maintains a constant relative-distance is that his proof analyzes the local testability of the tensor product using a result of Ben-Sasson and Sudan [BS06], which only holds for codes with high relative-distance. In our construction, on the other hand, we use a more recent result of Viderman [Vid15], which works for all distances. This allows us to work in the regime of sub-constant relative distance throughout the iterations.

- Second, the construction of [Mei09] uses a different method of distance amplification, due to [ABN$^+$92]. This method loses a constant factor in the rate regardless of the relative distance. Our construction, on the other hand, uses the AEL distance amplification. In the regime of sub-constant relative distance, the AEL method loses a factor of only $1 - o(1)$ in the rate in each iteration, as required.

We also note that the fact that we work in the regime of sub-constant relative distance, our construction suffers a larger blow-up in the query complexity in each iteration compared to [Mei09]. This is the reason why we obtain worse query complexity compared to [Mei09].

Another difference between the construction of [Mei09] and ours is that [Mei09] uses the random projection operation. The purpose of that operation in [Mei09] was to partially undo the rate loss

caused by tensoring — specifically, this operation caused the rate to decrease by a constant factor rather than being squared in each iteration. It would be interesting to see if one can use the random projection operation of [Mei09] to further improve our construction. However, it seems that the straightforward analysis of this operation does not work in the setting of high rate and sub-constant relative distance.

1.3 Additional results

Correctable *and* testable codes. Using our ideas , 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 affine-invariant 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 $n^{\widetilde{O}(1/\log\log n)}$ 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[2]. Since the LCCs we construct in this paper are linear, all our results apply to LDCs as well.

1.4 Organization of this paper

We review the required preliminaries in Section 2, construct our LCCs in Section 3, and construct our LTCs in Sections 4 and 5.

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.

[2]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.

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

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

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. ■

The tensor product of matrices. For a pair of matrices $G_1 \in \mathbb{F}^{m_1 \times n_1}$ and $G_2 \in \mathbb{F}^{m_2 \times n_2}$ their *tensor product* $G_1 \otimes G_2$ (a.k.a. the *Kronecker product*) is the $(m_1 \cdot m_2) \times (n_1 \cdot n_2)$ matrix over $\mathbb{F}$with entries

$$(G_1 \otimes G_2)_{(i_1,i_2),(j_1,j_2)} = (G_1)_{i_1,j_1} \cdot (G_2)_{i_2,j_2}$$

for every $i_1 \in [m_1]$, $i_2 \in [m_2]$, $j_1 \in [n_1]$ and $j_2 \in [n_2]$. The following is a well-known fact about the tensor product of matrices:

Fact 2.2. *Let A, B, C, D be matrices. Then,*

$$(A \otimes B) \cdot (C \otimes D) = (A \cdot C) \otimes (B \cdot D).$$

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. If $\Sigma = \mathbb{F}$, we simply say that C is linear. 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. The relative distance of C is the minimal fraction δ for which it holds that $\mathrm{dist}(c_1, c_2) \geq \delta$ for every pair of distinct codewords $c_1, c_2 \in C$. We will use the notation $\mathrm{dist}(z, C)$ to denote the relative distance of a string $z \in \Sigma^n$ from C, and say that z is ε-close (respectively, ε-far) to C if $\mathrm{dist}(z, C) < \varepsilon$ (respectively, if $\mathrm{dist}(z, C) \geq \varepsilon$).

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 in n that computes the encoding maps E_C of all the codes in the family. Let $C \subseteq \mathbb{F}^n$ be a linear code of dimension k. A *generating matrix* of C is an $n \times k$ matrix G such that the map $x \mapsto G \cdot x$ is an encoding map for C. 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 $\mathrm{dist}(c, z) \leq \tau$.

Concatenation. Concatenation is an operation on codes that can be used for reducing the alphabet size of a code. Let Λ and Σ be alphabets, where we think of Σ as being much larger than Λ. Let $C \subseteq \Sigma^n$ be a code over Σ and let $H \subseteq \Lambda^m$ be a code over Λ such that $|H| = |\Sigma|$. Let $\phi : \Sigma \to H$ be a bijection. The concatenation of C with H is the code $C' \subseteq \Lambda^{m \cdot n}$ that is obtained as follows: for each codeword $c \in C$, we construct a corresponding codeword $c' \in C'$ by replacing each symbol c_i with $\phi(c_i)$. We shall use the following well-known fact.

Fact 2.3 (Concatenation)**.** *Let $C \subseteq \Sigma^n$ be a code with rate r_C and relative distance δ_C, let $H \subseteq \Lambda^m$ be a code with rate r_H and relative distance δ_H, and let $C' \subseteq \Lambda^{m \cdot n}$ be the concatenation of C with H. Then C' has rate $r_C \cdot r_H$ and relative distance $\delta_C \cdot \delta_H$. Furthermore, if Λ is a field, C is Λ-linear, and H is linear, then C' is linear.*

Some useful codes. We use the following facts, which state the existence of Reed-Solomon codes, Gilbert-Varshamov codes, and Zyablov codes, and their relevant properties.

Fact 2.4 (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 $\mathrm{poly}(n, \log |\mathbb{F}|)$.*

Fact 2.5 (Gilbert-Varshamov codes [Gil52, Var57])**.** *For every $0 < r < 1$, there exists a (non-explicit) infinite family $\{GV_n\}_n$ of binary linear codes of with rate r and relative distance $H^{-1}(1-r)$, where H^{-1} is the inverse of the binary entropy function.*

The following codes, due to Zyablov, are obtained by concatenating the Reed-Solomon codes with the Gilbert-Varshamov codes.

Fact 2.6 (Zyablov bound [Zya71])**.** *For every $0 < r < 1$ and $\varepsilon > 0$, there exists an explicit infinite family $\{Z_n\}_n$ of binary linear codes of with rate r and relative distance*

$$\delta = \max_{r<R<1} \left\{ (1 - R - \varepsilon) \cdot H^{-1}\left(1 - \frac{r}{R}\right) \right\},$$

where H^{-1} is the inverse of the binary entropy function.

By choosing the parameters appropriately, we obtain the following useful special case.

Fact 2.7 (Special case of the Zyablov bound)**.** *For every sufficiently small $\eta > 0$, there exists an explicit infinite family $\{Z_n\}_n$ of binary linear codes of with rate $1 - \eta$ and relative distance at least η^3.*

Proof. Let $\eta > 0$ and let $r = 1 - \eta$. We apply Fact 2.6 with rate r and $\varepsilon = \eta/4$, thus obtaining an infinite family of codes $\{Z_n\}_n$. In order to lower bound the relative distance of $\{Z_n\}_n$, take $R = 1 - \eta/2$. Then, the relative distance of $\{Z_n\}_n$ is at least

$$\begin{aligned}
(1 - R - \varepsilon) \cdot H^{-1}\left(1 - \frac{r}{R}\right) &= \frac{\eta}{4} \cdot H^{-1}\left(1 - \frac{1-\eta}{1-\eta/2}\right) \\
&\geq \frac{\eta}{4} \cdot H^{-1}(\eta) \\
&\geq \eta^3,
\end{aligned}$$

where the last inequality holds for sufficiently small η. ■

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 2.8. 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 interested in the running time of the uniform local corrector.

Remark 2.9. 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, GS06] 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, where the main difference is that weak LTCs are required to reject only words which are sufficiently far from C, while strong LTCs are required to reject any word w not in C with probability proportional to the relative distance of w from C. From now on, we will work exclusively with strong LTCs, since it is a simpler notion and since it allows us to state a stronger result.

Definition 2.10. 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 $\mathrm{dist}(z, C)/4$.
- **Query complexity:** A makes at most q 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 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 $\rho = \frac{1}{4}$. However, given an LTC with $\rho < \frac{1}{4}$, it is possible to amplify ρ up to $\frac{1}{4}$ at the cost of increasing the query complexity. Hence, we chose to fix ρ to $\frac{1}{4}$ 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 $\frac{1}{\rho}$ times, and accepts if and only if all invocations of A accept. Clearly, this increases the query complexity by a factor of $\frac{1}{\rho}$ 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

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

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. The main object we shall use is a *sampler* defined below. Roughly speaking, a sampler is a bipartite d-regular graph in which the density of any subset T of right vertices can be approximated by the value of $E(s, T)/d$ for a uniformly random left vertex s.

Definition 2.11. Let $G = (U \cup V, E)$ be a bipartite d-regular graph with $|U| = |V| = n$. We say that G is an (α, γ)-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 2.12. *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 2.12 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 $\text{poly}(\log n)$) 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 a *bipartite double cover* G of G', that is, 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 $d \stackrel{\text{def}}{=} \text{poly}\left(\frac{1}{\alpha \cdot \gamma}\right)$ 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 our main theorem on LCCs, which immediately implies Theorem 1.3 from the introduction.

Theorem 1.3 (Main LCC theorem)**.** *For every $r\in(0,1)$ and $\varepsilon>0$, there exist a finite vector space Σ over $\mathbb{F}_2$ and an explicit infinite family of $\mathbb{F}_2$-linear codes $\{C_n\}_n$ over Σ satisfying:*

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

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

We explain how to construct our binary LCCs (Theorem 1.1) using Theorem 1.3 in Section 3.4 below.

The proof of Theorem 1.3 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 [AEL95, 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 (in a slightly non-trivial way). Those two steps are formalized in the following pair of lemmas, which are proved in Sections 3.1 and 3.2 respectively.

Lemma 3.1. *Suppose that there exists a code W that is locally correctable from τ_W fraction of errors with rate r_W and query complexity q. Then, for every $0<\tau<\frac{1}{2}$ and $\varepsilon>0$, 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 $p = \text{poly}(1/(\varepsilon \cdot \tau_W))$.*
- *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, τ, 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, τ, ε, Λ, and the block length of W, computes the local corrector of the corresponding code C. If the local corrector of W runs in time t_W, then the corresponding local corrector of C runs in time*

$$O(t_W) + q \cdot \text{poly}\left(1/(\varepsilon \cdot \tau_W), \log n_W\right),$$

  where n_W is the block length of W.
- *If W is $\mathbb{F}$-linear (where $\mathbb{F}$ is some finite field), then C is $\mathbb{F}$-linear.*

Lemma 3.2. *There exists an explicit infinite family of $\mathbb{F}_2$-linear codes $\{W_n\}_n$ satisfying:*

1. *The code 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. *The code 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 $\exp(\sqrt{\log n \cdot \log\log n})$.

Proof of Theorem 1.3. It is tempting to try to prove Theorem 1.3 by applying the distance amplication of Lemma 3.1 to the codes W_n of Lemma 3.2 with $\tau \approx \frac{1-r}{2}$. This would indeed yield codes of the required rate, relative distance and query complexity, but the alphabet size of those codes would be too large, and in particular, super-constant.

We therefore take a slightly indirect route: first, we apply the distance amplication of Lemma 3.1 to the codes W_n with $\tau \approx \varepsilon$. This yields codes with very high rate, constant (but small) relative distance, and alphabet of super-constant size. Then, we concatenate those codes with binary codes of high rate and small constant distance, thus obtaining *binary* codes with very high rate and small constant distance. Finally, we apply the distance amplication of Lemma 3.1 to the latter binary codes with $\tau \approx \frac{1-r}{2}$, and this gives the codes with the desired parameters. Details follow.

Fix a choice of the parameters r and ε. We describe how to construct the corresponding infinite family of codes $\{C_n\}_n$. We start by applying Lemma 3.1 to the family $\{W_n\}_n$ of Lemma 3.2 with $\tau_W = \Omega\left(\sqrt{\frac{\log\log n}{\log^3 n}}\right)$, $\tau = \frac{1}{64} \cdot \varepsilon$, and $\varepsilon = \frac{1}{32} \cdot \varepsilon$. This yields an infinite family of codes $\{W'_n\}_n$ that has rate $1 - \frac{1}{16} \cdot \varepsilon - \frac{1}{\log n} \geq 1 - \frac{1}{8} \cdot \varepsilon$ and alphabet size $\exp\left(\exp(\sqrt{\log n \cdot \log\log n})\right)$, and which is locally correctable from $\frac{1}{64} \cdot \varepsilon$ fraction of errors with query complexity $\exp(\sqrt{\log n \cdot \log\log n})$.

Let $\{Z_n\}_n$ be the infinite family of binary Zyablov codes of rate $1 - \frac{1}{8} \cdot \varepsilon$ and relative distance $\left(\frac{1}{8} \cdot \varepsilon\right)^3$ whose existence is guaranteed by Fact 2.7. We concatenate the family $\{W'_n\}_n$ with the family $\{Z_n\}_n$, thus obtaining an infinite family of *binary* linear codes $\{B_n\}_n$ with rate $1 - \frac{1}{4} \cdot \varepsilon$ and relative distance $\Omega(\varepsilon^4)$. Furthermore, it is not hard to see that those codes are locally correctable from $\Omega(\varepsilon^4)$ fraction of errors using query complexity $\exp(\sqrt{\log n \cdot \log\log n})$: the local corrector of $\{B_n\}_n$ emulates the local corrector of $\{W'_n\}_n$. Whenever the local corrector of $\{W'_n\}_n$ makes a query, the local corrector of $\{B_n\}_n$ reads the corresponding puported codeword of the inner Zyablov code, decodes it to the nearest codeword, and uses the result to answer the query of the local corrector of $\{W'_n\}_n$. It is easy to see that the query complexity of this local corrector is $\exp(\sqrt{\log n \cdot \log\log n})$, and standard arguments of coding theory show that it can correct $\Omega(\varepsilon^4)$ fraction of errors (see Section 3.4 for a sophisticated version of those arguments).

Finally, we apply Lemma 3.1 again, this time to the family $\{B_n\}_n$, with $\tau_W = \Omega\left(\varepsilon^4\right)$, $\varepsilon = \frac{1}{4} \cdot \varepsilon$, and

$$\tau = \frac{1}{2} \cdot \left(1 - \frac{r}{1 - \frac{1}{4} \cdot \varepsilon} - \frac{1}{4} \cdot \varepsilon\right) \geq \frac{1}{2} \cdot (1 - r - \varepsilon).$$

This results in an infinite family $\{C_n\}_n$ of $\mathbb{F}_2$-linear codes with rate

$$(1 - \frac{1}{4} \cdot \varepsilon) \cdot (1 - 2 \cdot \tau - \frac{1}{4} \cdot \varepsilon) = (1 - \frac{1}{4} \cdot \varepsilon) \cdot \left(1 - (1 - \frac{r}{1 - \frac{1}{4} \cdot \varepsilon} - \frac{1}{4} \cdot \varepsilon) - \frac{1}{4} \cdot \varepsilon\right) = r,$$

and alphabet size $\exp(\text{poly}(1/\varepsilon))$, which is locally correctable from $\tau \geq \frac{1-r-\varepsilon}{2}$ fraction of errors with query complexity

$$\exp(\sqrt{\log n \cdot \log\log n}) \cdot \text{poly}(1/\varepsilon) = \exp(\sqrt{\log n \cdot \log\log n}),$$

as required. The family $\{C_n\}_n$ is explicit with the required running time due to the first item in the "furthermore" part of Lemma 3.1, and has a uniform local corrector due to the second item of that part. ■

3.1 Proof of Lemma 3.1

3.1.1 Overview

Let $0 < \tau < \frac{1}{2}$. Our goal is to construct a code C that can be locally corrected from τ fraction of errors. 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 causes the errors to 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 3.1. Let $\{G_n\}_n$ be an infinite family of $(\tau_W, \frac{1}{2} \cdot \varepsilon)$-samplers as in Theorem 2.12, 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 block length 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

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

  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 $d \cdot t \leq d \log d = \text{poly}(1/(\varepsilon \cdot \tau_W))$. 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 3.1 by proving that C is locally correctable from τ fraction of errors with query complexity $\text{poly}(d) \cdot q$. 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 3.3. *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 $\mathrm{poly}(d) \cdot q$ queries to the oracle z.*

Before proving Lemma 3.3, 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 $B_1^c, \ldots, B_n^c$ and $S_1^c, \ldots, S_n^c$ be the corresponding blocks.

In order to decode c_i, the algorithm A should decode each of the symbols in the block $S_i^c \in \mathbb{F}^d$. 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_i^c belongs to one of the blocks $B_{j_1}^c, \ldots, B_{j_d}^c$, and therefore it suffices to retrieve the latter blocks. Now, each block $B_{j_h}^c$ 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 $B_{j_1}^c, \ldots, B_{j_d}^c$. 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 $B_{j_1}^c, \ldots, B_{j_d}^c$ correctly, and therefore computes the block S_i^c 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 3.3.

Proof of Lemma 3.3. 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, $l \stackrel{\mathrm{def}}{=} \lceil i_W/(b \cdot t) \rceil$.
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 $B_1^c, \ldots, B_n^c$ and $S_1^c, \ldots, S_n^c$ be the corresponding blocks. We also 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 $B_1^z, \ldots, B_n^z$ be the blocks that are obtained from $S_1^z, \ldots, S_n^z$ by reversing the permutation.
3. Define blocks $B_1^{z\prime}, \ldots, B_n^{z\prime}$ as follows: if B_i^z is $\left(\tau + \frac{\varepsilon}{2}\right)$-close to $RS_{b,d}$, then $B_i^{z\prime}$ is the nearest codeword of $RS_{b,d}$. Otherwise, $B_i^{z\prime}$ is the all-zeroes block.
4. Let $w^z \in \Lambda^{nW}$ be the string that is obtained by extracting the coordinates of w from each of the codewords $B_1^{z\prime}, \ldots, B_n^{z\prime}$.

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_l^z, it holds that B_l^z is $\left(\tau + \frac{\varepsilon}{2}\right)$-close to B_l^c.

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 $S_j^z \neq S_j^c$. 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 $B_{u_l}^z$ is $\left(\tau + \frac{\varepsilon}{2}\right)$-close to $B_{u_l}^c$, 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 3.2

For the proof of Lemma 3.2 we use the multiplicity codes of [KSY14], in a specialized sub-constant relative distance regime. We use the following results.

Lemma 3.4 ([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 $\text{poly}\left(\mathbb{F}^m, \binom{m+s-1}{m}\right)$ *time algorithm that computes an encoding map of M given s, d, m, and $\mathbb{F}$.*

Lemma 3.5 ([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 $O(s^m \cdot |\mathbb{F}|)$.*

Remark 3.6. As discussed in Section 4.3 of [KSY14], the local corrector of Lemma 3.5 can be implemented to have running time $\text{poly}(|\mathbb{F}|, s^m)$ 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 $|\mathbb{F}| \cdot O\left(\left(\frac{1}{\delta}\right)^m\right)$ queries, but this does not lead to any noticeable improvement for our setting.

We now prove Lemma 3.2.

Proof of Lemma 3.2. 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 3.5, 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 3.4, and the fact that it has an efficient uniform local corrector with the required running time follows from the discussion after Lemma 3.5. ■

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.

3.4 Binary LCCs that attain the Zyablov bound

In this section, we explain how to construct binary LCCs that attain the Zyablov bound, i.e., how to derive Theorem 1.1 from Theorem 1.3. Recall that for every rate $0 < r < 1$ and every $\varepsilon > 0$, we

would like to construct binary LCCs that can be decoded from

$$\frac{1}{2} \cdot Z_\varepsilon(r) = \frac{1}{2} \cdot \max_{r<R<1} \left\{ (1 - R - \varepsilon) \cdot H^{-1}(1 - \frac{r}{R}) \right\} \tag{1}$$

fraction of errors. We do it by concatenating the codes of Theorem 1.3 with an appropriate Gilbert-Varshamov codes, which is a standard idea in coding theory. More specifically, let $r < R < 1$ be the value that maximizes the expression in 1. We first obtain from Theorem 1.3 an infinite family of LCCs with rate R and relative distance $1 - R - \varepsilon$. We then concatenate the latter codes with the Gilbert-Varshamov codes of rate $\frac{r}{R}$ and relative distance $H^{-1}(1 - \frac{r}{R})$ (see Fact 2.5). The resulting concatenated codes have rate r and relative distance

$$\max_{r<R<1} \left\{ (1 - R - \varepsilon) \cdot H^{-1}(1 - \frac{r}{R}) \right\}.$$

However, we still need to show that they can be locally corrected from half the relative distance. In order to show it, we prove a variant of the GMD decoding of [For66] that is tailored for locally-correctable codes. To this end, we use the following definitions, which generalize local correction to deal with both errors and erasures.

Definition 3.7. Let $C \subseteq \Sigma^n$ be a code, and let $z \in (\Sigma \cup \{?\})^n$. The fraction of errors in z with respect to a codeword $c \in C$ is the fraction of coordinates i such that $z_i \notin \{c_i, ?\}$. The fraction of erasures in z is the fraction of coordinates i such that $z_i =?$.

Definition 3.8. We say that a code $C \subseteq \Sigma^n$ is locally correctable from δ fraction of twice-errors and erasures 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 $z \in (\Sigma \cup \{?\})^n$ that has ρ^e fraction of errors and $\rho^?$ fraction of erasures with respect to some codeword $c \in C$, such that
$$2 \cdot \rho^e + \rho^? \leq \delta.$$
- **Output:** A outputs c_i with probability at least $\frac{2}{3}$.
- **Query complexity:** A makes at most q queries to the oracle z.

Note that the fraction of errors in the above condition is multiplied by 2. Specifically, observe that if C is locally correctable from δ fraction of twice-errors and erasures then in particular it is locally correctable from $\delta/2$ fraction of errors (rather than δ fraction of errors). We now have the following variant of GMD decoding.

Theorem 3.9 (GMD decoding of LCCs)**.** *Let $C \subseteq \Sigma^n$ be a locally-correctable code from δ_C fraction of twice-errors and erasures with query complexity q. Let $H \subseteq \Lambda^m$ be a code with relative distance δ_H such that $|H| = |\Sigma|$. Let $C' \subseteq \Lambda^{m \cdot n}$ be the concatenation of C with H. Then, C' is locally correctable from*

$$\frac{1}{2} \cdot \delta_C \cdot \delta_H - O(\frac{1}{\sqrt{n}})$$

fraction of errors with query complexity $O(q \cdot m)$.

Furthermore, there is an oracle algorithm that when given

- *black box access to a local corrector of C that runs in time t_C, and*
- *black box access to an algorithm that decodes H from $\delta_H/2$ fraction of errors in time t_H,*

computes a local corrector of the corresponding code C' that runs in time $t_C + O(q \cdot t_H) + \text{poly}(m, \log n)$.

We prove Theorem 3.9 in Section 3.4.1 below. It remains to explain how to use Theorem 3.9 to show that the codes we constructed above can be locally corrected from the fraction of errors in Equation 1. Basically, this follows by applying Theorem 3.9 with C being the codes obtained from Theorem 1.3, with H being the Gilbert-Varshamov codes, and with[3] $\delta_C = 1 - R - \varepsilon$, $\delta_H = H^{-1}(1 - \frac{r}{R})$. However, in order to apply Theorem 3.9, we need to show that the codes obtained from Theorem 1.3 can be locally corrected from $1 - r - \varepsilon$ fraction of *twice-errors and erasures*. Note that Theorem 1.3 only tells us that they can be locally corrected from $\frac{1-r-\varepsilon}{2}$ fraction of *errors*.

It remains to show that the codes obtained from Theorem 1.3 can be locally corrected from twice-errors and erasures. To this end, we observe that the AEL distance-amplification (Lemma 3.1) yields codes that are locally correctable from twice-errors and erasures. More formally, we have the following result.

Lemma 3.10 (Lemma 3.1 for twice-errors and erasures)**.** *Let W, τ_W be as in Lemma 3.1, and let $0 < \delta < 1$ and $\varepsilon > 0$. Then, there exists a code C that is locally correctable from δ fraction of twice-errors and erasures with query complexity $q \cdot \text{poly}(1/(\varepsilon \cdot \tau_W))$ that has rate $r_W \cdot (1 - \delta - \varepsilon)$. Furthermore, C satisfies analogous properties of the code C in Lemma 3.1.*

If we replace Lemma 3.1 with Lemma 3.10 in the proof of Theorem 1.3, we immediately obtain that the codes of Theorem 1.3 can be locally corrected from the required fraction of twice-errors and erasures, as required.

Proof sketch of Lemma 3.10. This lemma is proved exactly as Lemma 3.1, with the following modification. Recall that the key idea in the analysis of the local corrector of C in that lemma was the following: when the local corrector is given oracle access to a corrupted codeword, almost all the blocks $B_1, \ldots, B_n$ contain the "correct" fraction of errors (up to an additive term of $\Omega(\varepsilon)$), and therefore those blocks can be corrected by the decoding algorithm of Reed-Solomon codes. The same argument can be used to show that if the corrupted codeword contains both errors and erasures, then almost all the blocks $B_1, \ldots, B_n$ again contain the "correct" fraction of twice-errors and erasures (again, up to an additive term of $\Omega(\varepsilon)$). Thus, those blocks can again be corrected by the decoding algorithm of Reed-Solomon codes, which is well-known to decode from twice-errors and erasures. ■

3.4.1 Proof of Theorem 3.9

Comparison with GMD decoding. The proof of Theorem 3.9 follows the proof of the GMD decoding of [For66], with small modifications. Recall that the standard presentation of the GMD algorithm (e.g., [Sud01, Lecture 11]) is as follows: First, a randomized decoding algorithm is presented. Then, it is shown that the latter algorithm succeeds *in expectation* (rather than *with high probability*). Finally, the randomized algorithm is *derandomized*, yielding a deterministic algorithm that always succeeds.

[3] The $O(\frac{1}{\sqrt{n}})$ term of Theorem 3.9 can be incorporated within the parameter ε.

In our setting, we do not know how to perform the derandomization step. The reason is that the derandomization is done by enumerating over all the possible sequence of coin tosses of the randomized algorithm, and checking for each sequence whether it yields the correct answer — i.e., whether it yields a codeword that is sufficiently close to the received string. In our case, we do know how to check that the string that the decoder outputs is a valid codeword, since we are only allowed to query a small part of this string.

In order to resolve this issue, we skip the derandomization step, and instead observe that the original randomized decoder actually succeeds with high probability. Details follow.

The algorithm. Let $C \subseteq \Sigma^n$ be locally-correctable code from δ_C fraction of twice errors and erasures with query complexity q. Let $H \subseteq \Lambda^m$ be a code with relative distance δ_H such that $|H| = |\Sigma|$. Let $C' \subseteq \Lambda^{m \cdot n}$ be the concatenation of C with H, and let $\phi : \Sigma \to H$ be the bijection that is used to define the concatenation. Let A be the local corrector of C.

In what follows, we assume that A has success probability at least $\frac{7}{9}$ (rather than $\frac{2}{3}$). This can be guaranteed by amplification, while increasing the query complexity of A by at most a constant factor. We also assume that the code H has some associated decoding algorithm. If no such algorithm is provided, we use the brute-force algorithm that enumerates over all codewords of H and outputs the codeword h that is closest to v.

We describe a local corrector A' for C'. When A' is given access to a string $z' \in \Lambda^{m \cdot n}$, and a coordinate $i' \in [m \cdot n]$, it behaves as follows. First, A' finds the index $i \in [n]$ of the block to which i' belongs (formally, $i \stackrel{\text{def}}{=} \lceil i'/m \rceil$). Then, A' emulates the action of the local corrector A of C on the coordinate i. Whenever A makes a query to a coordinate $j \in [n]$, the local corrector A' emulates the query as follows:

1. A' queries the block of z' that corresponds to j, that is, it queries the coordinates $(j-1) \cdot m + 1, \ldots, (j-1) \cdot m + m$ of z'. Let $v \in \Lambda^m$ be the obtained string.

2. A' invokes the decoding algorithm of H on v. Let h be the obtained codeword of H (if the decoding fails, we set h to be an arbitrary codeword of H).

3. With probability
$$\min\left\{1, \operatorname{dist}(v,h)/\frac{\delta_H}{2}\right\},$$
the local corrector A' answers the query of A with $\phi^{-1}(h)$ (i.e., the symbol in Σ that corresponds to h).

4. Otherwise, it answers the query with an erasure, i.e., with the symbol "?".

When the emulation of A ends, A outputs a symbol $\sigma \in \Sigma$. Then, the local corrector A' finishes and outputs the symbol of $\phi(\sigma)$ that corresponds to i'.

The analysis. It is easy to see that the query complexity of A' is m times the query complexity of A. This means that the query complexity of A' is at most $O(m \cdot q)$, as required (there is a constant factor here because we amplified the success probability of A). It is also not hard to see that A' has the required running time.

It remains to prove that if there is a codeword $c' \in C'$ such that

$$\mathrm{dist}(z', c') \leq \frac{1}{2} \cdot \delta_C \cdot \delta_H - O(\frac{1}{\sqrt{n}}),$$

then A' outputs c'_i with probability at least $\frac{2}{3}$. To this end, observe that the output of A' is distributed exactly like the output of the following algorithm A'' (which is not query efficient). The algorithm A'' takes the string z' and the coordinate i', and performs the following steps:

1. The algorithm A'' constructs a string $z \in \Sigma^n$ by setting each coordinate $j \in [n]$ according to the way that A' emulates a query of A to j. More specifically, A'' sets z_j as follows:
   (a) A'' queries the block of z' that corresponds to j, thus obtaining a string $v \in \Lambda^m$.
   (b) A'' invokes the decoding algorithm of H on v, thus obtaining a codeword $h \in H$.
   (c) With probability
   $$\min\left\{1, \mathrm{dist}(v,h)/\frac{\delta_H}{2}\right\},$$
   the algorithm sets $z_j = \phi^{-1}(h)$, and otherwise it sets $z_j = ?$.
2. The algorithm A'' invokes the local corrector A on z and the coordinate $i \stackrel{\mathrm{def}}{=} \lceil i'/m \rceil$, thus obtaining a symbol $\sigma \in \Sigma$.
3. The algorithm A'' outputs $\phi(\sigma)_{i' \bmod m}$.

It therefore suffices to prove that A'' outputs c'_i with probability at least $\frac{2}{3}$. To this end, we prove the following result.

Proposition 3.11. *Let c be the codeword of C that corresponds to c'. Let ρ^e and $\rho^?$ be the fractions of errors and erasures in z with respect to c respectively. Then, with probability at least $\frac{8}{9}$ it holds that*

$$2 \cdot \rho^e + \rho^? \leq \delta_C. \tag{2}$$

Before proving Proposition 3.11, observe that it implies the required result. Indeed, whenever Inequality 2 holds, the local corrector A (and thus the algorithm A'') is guaranteed to succeed with probability at least $\frac{7}{9}$. The proposition says that the probability that this inequality does not hold is at most $\frac{1}{9}$, so it follows that A'' succeeds with probability at least $\frac{7}{9} - \frac{1}{9} = \frac{2}{3}$, as required.

Proof of Proposition 3.11. For every $j \in [n]$, let I^e_j be the indicator random variable of the event that z_j is an error (i.e., $z_j \notin \{c_j, ?\}$), and let $I^?_j$ be the indicator random variable of the event that z_j is an erasure. For every $j \in [n]$, define the random variable

$$T_j \stackrel{\mathrm{def}}{=} 2 \cdot I^e_j + I^?_j.$$

We wish to prove that

$$\frac{1}{n} \cdot \sum_{j=1}^{n} T_j \leq \delta_C$$

with probability at least $\frac{8}{9}$. It is a known fact[4] about GMD decoding that

$$\mathbb{E}\left[\frac{1}{n}\cdot\sum_{j=1}^{n}T_j\right] \leq \delta_C - O(\frac{1}{\sqrt{n}})$$

(see, e.g., [Sud01, Lecture 11]). Thus, it remains to prove that the average $\frac{1}{n}\cdot\sum_{j=1}^{n}T_j$ is concentrated around its expectation. To this end, observe the random variables T_j are independent, and that each T_j takes a value in $[0,2]$. Therefore a standard application of the Hoeffding bound implies that

$$\Pr\left[\frac{1}{n}\cdot\sum_{j=1}^{n}T_j \leq \delta_C - O(\frac{1}{\sqrt{n}}) + \varepsilon\right] \geq 1 - e^{-\frac{1}{2}\cdot\varepsilon^2\cdot n}.$$

In particular, by setting ε to be equal to the $O(\frac{1}{\sqrt{n}})$ term, and choosing the constant in the Big-O appropriately, we get that

$$\Pr\left[\frac{1}{n}\cdot\sum_{j=1}^{n}T_j \leq \delta_C\right] \geq \frac{8}{9},$$

as required. ■

Remark 3.12. In the proof above, we argued that the outputs of A' and A'' are identically distributed. Actually, this only holds under the assumption that A never makes the same query twice. However, this can be assumed without loss of generality.

4 LTCs with quasi-polylogarithmic query complexity

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

Theorem 1.4 (Main LTC theorem)**.** *For every $r \in (0,1)$, and $\varepsilon > 0$, there exist a finite vector space Σ over $\mathbb{F}_2$ and an explicit infinite family of $\mathbb{F}_2$-linear codes $\{C_n\}_n$ over Σ satisfying:*

- *C_n has block length n, rate at least r, and relative distance at least $1 - r - \varepsilon$.*
- *C_n is locally testable with query complexity $(\log n)^{O(\log\log n)}$.*
- *The size of Σ is at most $\exp(\mathrm{poly}(1/\varepsilon))$.*

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

We explain how to construct our binary LTCs (Theorem 1.2) using Theorem 1.4 in Section 4.2 below.

The proof of Theorem 1.4 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 [AEL95, AL96]. In the second step, we construct a locally-testable code W_n with the desired query complexity but

[4] Actually, the known fact is that if $\mathrm{dist}(c', C') \leq \frac{1}{2}\cdot\delta_C\cdot\delta_H$ then $\mathbb{E}\left[\frac{1}{n}\cdot\sum_{j=1}^{n}T_j\right] \leq \delta_C$. However, we can get the desired inequality by substracting the term $O(\frac{1}{\sqrt{n}})$ in the appropriate places in the proof of this fact.

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. The first lemma is proved in Section 4.1, while the proof of the second lemma is deferred to Section 5. The proof of Theorem 1.4 from Lemmas 4.1 and 4.2 is analogous to the proof of Theorem 1.3 from Lemmas 3.1 and 3.2, and is given below.

Lemma 4.1. *Suppose that there exists a code W with relative distance δ_W that is locally testable with rate r_W and query complexity q. 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 $p = \mathrm{poly}(1/(\varepsilon \cdot \delta_W))$.*

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, δ, 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, δ, ε, Λ, and the block length of W, computes the local tester of the corresponding code C. If the local tester of W runs in time t_W, then the corresponding local tester of C runs in time*

$$O(t_W) + q \cdot \mathrm{poly}\left(1/(\varepsilon \cdot \tau_W), \log n_W\right),$$

  where n_W is the block length of W.
- *If W is $\mathbb{F}$-linear (where $\mathbb{F}$ is some finite field), then C is $\mathbb{F}$-linear.*

Lemma 4.2. *There exists an explicit infinite family of binary 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 $1/\mathrm{polylog}(n)$.*
2. *W_n is locally testable with query complexity $(\log n)^{O(\log\log n)}$.*

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

Proof of Theorem 1.4. We would like to construct the desired LTCs by amplifying the relative distance of the LTCs of Lemma 4.2. The straightforward solution would be to apply the AEL distance-amplification (Lemma 4.1) to those codes and amplify them directly to the desired distance. However, this solution is problematic, since Lemma 4.1 yields codes whose alphabet size depends on the original relative distance. In our case, this would yield super-constant alphabet size, while we would like our codes to have constant alphabet size.

In order to resolve this issue, we construct our LTCs in a multiple steps:

We start by amplifying the LTCs of Lemma 4.2 only to a small relative distance $\mathrm{poly}(\varepsilon)$. This yields codes with super-constant alphabet size, and only decreases the rate by a factor of $1 - \mathrm{poly}(\varepsilon)$.

Next, we concatenate the latter codes with some binary codes with relative distance $\mathrm{poly}(\varepsilon)$. This yields binary codes with relative distance $\mathrm{poly}(\varepsilon)$, and again only decreases the rate by a factor of $1 - \mathrm{poly}(\varepsilon)$.

Finally, we amplify the latter LTCs to the desired relative distance. This yields codes whose alphabet size depends only on ε, as required.

We now provide the formal details. Fix $0 < r < 1$ and $\varepsilon > 0$, and let $\delta = 1 - r - \varepsilon$ be the desired relative distance. Let $\{W_n\}_n$ be the infinite family of Lemma 4.2. We start by applying Lemma 4.1 with parameters $\delta' = \frac{1}{5} \cdot \varepsilon$ and $\varepsilon = \frac{1}{5} \cdot \varepsilon$. This yields an infinite family of $\mathbb{F}_2$-linear LTCs $\{U_n\}_n$ with relative distance $\frac{1}{5} \cdot \varepsilon$, rate at least

$$1 - \frac{2}{5} \cdot \varepsilon - O(\frac{1}{\log n}) \geq 1 - \frac{3}{5} \cdot \varepsilon,$$

query complexity $(\log n)^{O(\log \log n)}$, and alphabet size $\exp\left((\log n)^{O(\log \log n)}\right)$.

Next, let $\{Z_n\}_n$ be an infinite family of binary Zyablov codes (Fact 2.7) with relative distance $\left(\frac{1}{5} \cdot \varepsilon\right)^3$ and rate $1 - \frac{1}{5} \cdot \varepsilon$. We concatenate the family $\{U_n\}_n$ with the family $\{Z_n\}_n$, thus obtaining an infinite family of binary LTCs $\{V_n\}_n$ with relative distance $\Omega(\varepsilon^4)$, rate at least $1 - \frac{4}{5} \cdot \varepsilon$ and query complexity $(\log n)^{O(\log \log n)}$. The upper bound on the query complexity can be proved by noting that concatenation is a block-wise operation (as in Definition 5.7), and thus we can apply Proposition 5.8.

Finally, we apply Lemma 4.1 to the family $\{V_n\}_n$ with $\delta' = \delta$, $\varepsilon = \frac{1}{5} \cdot \varepsilon$, thus obtaining an infinite family of binary codes $\{C_n\}_n$ with relative distance δ, rate

$$(1 - \delta - \frac{1}{5} \cdot \varepsilon) \cdot (1 - \frac{4}{5} \cdot \varepsilon) = (r + \varepsilon - \frac{1}{5} \cdot \varepsilon) \cdot (1 - \frac{4}{5} \cdot \varepsilon) \geq r,$$

query complexity $(\log n)^{O(\log \log n)}$, and alphabet size $\exp(\mathrm{poly}(1/\varepsilon))$, as required. It can be verified that the family $\{C_n\}_n$ is linear, explicit, and has a uniform local tester with the desired running time. ■

4.1 Proof of Lemma 4.1

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 $q \cdot \mathrm{poly}(1/(\varepsilon \cdot \delta_W))$. 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, $l \stackrel{\mathrm{def}}{=} \lceil j/(b \cdot t) \rceil$.

[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 3.1, 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 $\frac{1}{4} \cdot \mathrm{dist}(z, C)$. To this end, it suffices to prove that A rejects with probability at least $\frac{1}{\mathrm{poly}(d)} \cdot \mathrm{dist}(z, C)$ — as explained in Section 2.3, this rejection probability can be amplified to $\frac{1}{4} \cdot \mathrm{dist}(z, C)$ while increasing the query complexity by a factor of $\mathrm{poly}(d)$, 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 the end of this section.

Proposition 4.3. *A'_W rejects a string $y \in (\Lambda \cup \{?\})^{n_W}$ with probability at least $\frac{1}{8} \cdot \mathrm{dist}(y, W)$.*

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

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

and therefore

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

It thus follows that A rejects z with probability at least

$$\frac{1}{8} \cdot \mathrm{dist}(w^z, W) \geq \frac{1}{\mathrm{poly}(d)} \cdot \mathrm{dist}(z, C),$$

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. We turn to proving Proposition 4.3.

Proof of Proposition 4.3. We may assume without loss of generality that A_W makes at least one query that is uniformly distributed over $[n_W]$: otherwise, we can change A_W such that it makes an additional uniformly distributed query and ignores the answer. This assumption means that A'_W makes at least one query that is uniformly distributed over $[n_W]$, and rejects if the answer is "?". Let

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

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

- E **is "large":** Suppose that $\frac{|E|}{n_W} \geq \frac{1}{2} \cdot \text{dist}(y, W)$. In this case, the uniformly distributed query of A'_W hits a coordinate in E with probability at least $\frac{1}{2} \cdot \text{dist}(y, W)$. In such a case, A'_W rejects, and the proposition follows.

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

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

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

  $$\begin{aligned}
  \Pr\left[A'_W \text{ rejects } y\right] &= \Pr[\mathcal{E}] \cdot \Pr\left[A'_W \text{ rejects } y | \mathcal{E}\right] + \Pr[\neg\mathcal{E}] \cdot \Pr\left[A'_W \text{ rejects } y | \neg\mathcal{E}\right] \\
  &= \Pr[\mathcal{E}] \cdot 1 + \Pr[\neg\mathcal{E}] \cdot \Pr[A_W \text{ rejects } y_0 | \neg\mathcal{E}] \\
  &\geq \Pr[\mathcal{E}] \cdot \Pr[A_W \text{ rejects } y_0 | \mathcal{E}] + \Pr[\neg\mathcal{E}] \cdot \Pr[A_W \text{ rejects } y_0 | \neg\mathcal{E}] \\
  &= \Pr[A_W \text{ rejects } y_0] \\
  &\geq \frac{1}{4} \cdot \text{dist}(y_0, W) \\
  &\geq \frac{1}{8} \cdot \text{dist}(y, W),
  \end{aligned}$$

  as required.

This concludes the proof. ■

4.2 LTCs that attain the Zyablov bound

In this section, we explain how to obtain our binary LTCs from the above LTCs, i.e., how to derive Theorem 1.2 from Theorem 1.4. As in the case of LCCs (Section 3.4), we construct our binary LTCs by concatenating the codes $\{C_n\}_n$ of Theorem 1.4 with the Gilbert-Varshamov codes $\{GV_n\}_n$ (Fact 2.5). Let us denote the resulting family of concatenated codes by $\{C'_n\}_n$. The analysis of the rate and relative distance of these codes is the same as in the case of LCCs.

It remains to show that these codes are locally testable. Let A be the tester of the codes $\{C_n\}_n$ of Theorem 1.4, and let us denote by Σ the alphabet of those codes. We describe a tester A' for

the concatenated codes $\{C'_n\}_n$: The tester A' emulates the tester A. Whenever A makes a query, the tester A' reads the corresponding block and verifies that this block is a legal codeword of the Gilbert-Varshamov code. If it is a legal codeword, A' retrieves from it the corresponding symbol of Σ and feeds it as an answer to the query of A. Otherwise, A' rejects. If the emulation of A finishes, the tester A' accepts if A accepts and rejects otherwise.

The tester A' has query complexity that is at most a constant times the query complexity of A, since the inner codes have constant block length. It is also easy to see that A' satisfies the completeness property, i.e., it accepts codewords of $\{C'_n\}_n$ with probability 1. It remains to prove that A' rejects a string z' with probability that is proportional to its distance from the corresponding code $C'_{|z'|}$. To this end, observe that A' can be viewed as emulating the running of A on the string z that is obtained by replacing blocks of z' that are legal codewords with the corresponding symbols in Σ, and replacing the other blocks with the symbol "?". The required bound on the rejection probability now follows from Proposition 4.3 using a similar argument to that of Section 4.1.

5 LTCs with quasi-polylogarithmic query complexity and sub-constant distance

In this section we prove Lemma 4.2, restated below.

Lemma (4.2)**.** *There exists an explicit infinite family of binary 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 $1/\operatorname{polylog}(n)$.*
2. *W_n is locally testable with query complexity $(\log n)^{O(\log\log n)}$.*

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

In Sections 5.1 and 5.2 we describe the two basic operations on codes that are used for the proof of the above lemma: the tensor product and a special type of distance amplification that we denote as $\square$-distance amplification. Then in Section 5.3 we describe the construction of the codes $\{W_n\}_n$.

5.1 Tensor product

The tensor product of codes is an operation whose usefulness for constructions of LTCs has been discovered by [BS06], and was studied further in [Val05, CR05, DSW06, Mei09, BV09, BV15, Vid12, GM12, Mei12, Vid13, Vid15]. For a linear code $C_1 \subseteq \mathbb{F}^{n_1}$ and $C_2 \subseteq \mathbb{F}^{n_2}$, their *tensor product code* $C_1 \otimes C_2 \subseteq \mathbb{F}^{n_1 \times n_2}$ consists of all the matrices M such that all the rows of M are codewords of C_2 and all the columns are codewords of C_1. For a linear code C, let $C^1 \stackrel{\text{def}}{=} C$ and $C^m \stackrel{\text{def}}{=} C^{m-1} \otimes C$. The following are some useful facts regarding the tensor product operation and its effect on the classical parameters of a code (see e.g. [Sud01, DSW06]).

Fact 5.1 (Properties of tensor product)**.** *Let $C_1 \subseteq \mathbb{F}^{n_1}$ and $C_2 \subseteq \mathbb{F}^{n_2}$ be linear codes of rates r_1, r_2 and relative distances δ_1, δ_2 respectively. Then $C_1 \otimes C_2 \subseteq \mathbb{F}^{n_1 \times n_2}$ is a linear code of rate $r_1 \cdot r_2$ and relative distance $\delta_1 \cdot \delta_2$. Furthermore, if G_1, G_2 are generating matrices of C_1, C_2 respectively, then the tensor product $G_1 \otimes G_2$ is a generating matrix of $C_1 \otimes C_2$ (see Section 2 for the definition of $G_1 \otimes G_2$).*

In our construction, we would like to apply the tensor product operation to LTCs. In order for the argument to work, we need the tensor product to preserve the local testability of the codes. To this end, we define a property of codes called "property $\square$", and observe that the tensor product operation preserves the local testability of codes that have that property. Then, throughout our construction of the codes $\{W_n\}_n$, we will make sure that the codes we work with have the property $\square$.

Definition 5.2 (Property $\square$)**.** We say that a code C has property $\square$ if and only if there exists a code D such that $C = D^2$.

Let $C \in \mathbb{F}^n$ be a linear code, and consider the code C^4. The codewords of C^4 are of length n^4, and it is useful to identify their coordinates set with the 4-dimensional hypercube $[n]^4$. We say that a set $P \subseteq [n]^4$ is an *axis-parallel plane* of the hypercube $[n]^4$ if there exist $\alpha, \beta \in [n]$ and $k_1 \neq k_2 \in [4]$ such that

$$P = \left\{ (i_1, i_2, i_3, i_4) \in [n]^4 : i_{k_1} = \alpha, i_{k_2} = \beta \right\}.$$

The following fact gives an alternative characterization of C^4, and follows from our definition of tensor product codes.

Fact 5.3. *Let $C \in \mathbb{F}^n$ be a linear code, and let $z \in \mathbb{F}^{n^4}$. Then, $z \in C^4$ if and only if $z|_P \in C^2$ for every axis-parallel plane P.*

It turns out that the characterization in Fact 5.3 is *robust*, in the sense that if, for the average axis-parallel plane P, it holds that $z|_P$ is close to C^2, then z is close to C^4. Conversely, if z is far from C^4, then $z|_P$ is far from C^2 on average. This was proved by [BS06, Vid15] for the purpose of constructing locally-testable codes using tensor products. In particular, we use the following result.

Theorem 5.4 ([Vid15, Theorem 4.4])**.** *Let $C \subseteq \mathbb{F}^n$ be a code with relative distance δ. Let $z \in \mathbb{F}^{n^4}$ be a string, and let $P \subseteq [n]^4$ be a random axis-parallel plane. Then,*

$$\mathbb{E}\left[\operatorname{dist}(z|_P, C^2)\right] \geq \frac{1}{300} \cdot \delta^{12} \cdot \operatorname{dist}(z, C^4).$$

We use the latter theorem for showing that the tensor product operation maintains the local testability of codes:

Corollary 5.5 (Local testability of tensor product)**.** *Let C be a code of relative distance δ that is locally testable with query complexity q and running time T. Suppose furthermore that C has the property $\square$, i.e., that there exists a code D such that $C = D^2$. Then C^2 is locally testable with query complexity $1200 \cdot q/\delta^6$ and running time $(O(T) + \operatorname{poly}\log(n))/\delta^6$.*

Proof. Suppose that C is a code over $\mathbb{F}$ of block length n, and let $n' \stackrel{\text{def}}{=} \sqrt{n}$. Observe that $D \subseteq \mathbb{F}^{n'}$ is a code of relative distance $\delta' \stackrel{\text{def}}{=} \sqrt{\delta}$, and that $C^2 = D^4$. Let A be the local tester of C. We describe a local tester A' for C^2.

When given a received word $z \in \mathbb{F}^{n^2} = \mathbb{F}^{(n')^4}$, the tester A' chooses a random axis-parallel plane $P \subseteq [n']^4$, runs the local tester A to verify that $z|_P \in C = D^2$, and accepts if and only if A

accepts. Clearly, A' uses q queries, runs in time $O(T) + \text{poly}\log(n)$, and accepts codewords of C^2 with probability 1. We show that A' rejects a string $z \in \mathbb{F}^{n^2}$ with probability at least

$$\frac{1}{1200} \cdot \delta^6 \cdot \text{dist}(z, C^2).$$

This will imply the required result, since the latter rejection probability can be amplified to $\frac{1}{4} \cdot \text{dist}(z, C^4)$ by increasing the query complexity and running time by a factor of $1200/\delta^6$ (see the discussion in Section 2.3).

Let $z \in \mathbb{F}^{n^2} = \mathbb{F}^{(n')^4}$ be a string. Then, for every axis-parallel plane $P \subseteq [n]^4$, the local tester A rejects $z|_P$ with probability at least $\frac{1}{4} \cdot \text{dist}(z|_P, C^2)$. By Theorem 5.4, it follows that A' rejects z with probability at least

$$\begin{aligned}
\mathbb{E}_P\left[\frac{1}{4} \cdot \text{dist}\left(z|_P, D^2\right)\right] &= \frac{1}{4} \cdot \mathbb{E}_P\left[\text{dist}\left(z|_P, D^2\right)\right] \\
&\geq \frac{1}{4} \cdot \frac{1}{300} \cdot (\delta')^{12} \cdot \text{dist}(z, D^4) \\
&= \frac{1}{1200} \cdot \delta^6 \cdot \text{dist}(z, C^2),
\end{aligned}$$

as required. ■

5.2 □-distance amplification

The second operation we use is based on the AEL distance amplification method [AEL95, AL96], but is designed to improve the relative distance of an LTC C while preserving the property □. This is done roughly as follows: Let C be an LTC that has the property □, i.e., there exists some code D such that $C = D^2$. We would like to improve the relative distance of C. To this end, we apply the AEL distance-amplification technique to D, thus obtaining a code D', and set $C' = (D')^2$ to be the new code. Clearly, C' has the property □, and it is not hard to show that this operation improves the relative distance. The main challenge in this section will be to show that C' is still locally testable.

There is also another issue that needs to be handled: the AEL distance-amplification technique increases the alphabet size of the code. Thus, if D was a binary linear code, the code D' would have alphabet $\Sigma = \{0,1\}^p$ for some $p \in \mathbb{N}$. In particular, D' would not be a linear code, but only an $\mathbb{F}_2$-linear code. This is problematic, both because we need to maintain the linearity of our codes (as otherwise the tensor product operation would not be defined), and because we do not want the alphabet size of our codes to increase throughout the iterations. In order to resolve this issue, after applying the AEL amplification, we concatenate the resulting code with a binary inner code to reduce the alphabet size back to 2.

The following lemma states the □-distance amplification. We only state the amplification for binary linear codes and for sufficiently small distances, which is sufficient for our purposes.

Lemma 5.6 (Properties of □-distance amplification)**.** *There exists a universal constant $\delta_0 > 0$ such that the following holds. Let $C \subseteq \{0,1\}^n$ be a binary linear code with relative distance δ and rate r that is locally testable with query complexity q. Suppose further that there exists a code D such that $C = D^2$. Then, for every sufficiently small δ' such that $\delta_0 > \delta' > \delta$, there exists a binary linear code C' with relative distance δ' that is locally testable with query complexity $q \cdot \text{poly}(1/\delta)$ such that:*

- $|C'| = |C|$.
- *C' has rate at least* $\left(1 - 6 \cdot \sqrt[12]{\delta'}\right) \cdot r$.
- *There exists a code D' such that $C' = (D')^2$.*

Furthermore,

- *There is a polynomial time algorithm that computes a bijection from every code C to the corresponding code C', given r, δ, δ', and the generating matrix of D.*
- *There is an oracle algorithm that computes the local tester of the corresponding code C' when given black box access to the local tester of C, and given also r, δ, δ', and the block length of C. The resulting local tester of C' runs in time* $\mathrm{poly}(\log n/\delta) \cdot T_C$ *where T_C is the running time of the local tester of C.*

We now construct the code C' and show that it has the required rate and relative distance. In the rest of the section we will show that C' is locally testable with the required query complexity and running time.

As explained above, the basic idea of the construction is to first apply the AEL distance-amplification to D, and then concatenate the resulting code with a binary inner code to reduce the alphabet size back to 2. We choose δ_0 to be sufficiently small such that Fact 2.7 holds (the special case of the Zyablov bound). The code C' is constructed as follows:

- Recall that the code D has rate $\sqrt{r}$ and relative distance $\sqrt{\delta}$. We apply the AEL distance-amplification (Lemma 4.1) to the code D, choosing both the parameters δ' and ε to be $\sqrt[4]{\delta'}$. This results in an $\mathbb{F}_2$-linear code $\tilde{D}$ with relative distance $\sqrt[4]{\delta'}$ and rate $(1 - 2 \cdot \sqrt[4]{\delta'}) \cdot \sqrt{r}$ over the alphabet $\Sigma = \{0,1\}^p$ (for some $p = \mathrm{poly}(1/\delta)$).
- We choose our inner code to be the binary linear code Z obtained from the special case of the Zyablov bound (Fact 2.7)[6] with relative distance $\delta_Z \stackrel{\text{def}}{=} \sqrt[4]{\delta'}$ and relative distance at least $r_Z \stackrel{\text{def}}{=} 1 - \sqrt[12]{\delta'}$. We now concatenate $\tilde{D}$ with the code Z, thus obtaining a binary linear code D' relative distance $\sqrt[4]{\delta'} \cdot \delta_Z = \sqrt{\delta'}$ and rate at least
$$(1 - 2 \cdot \sqrt[4]{\delta'}) \cdot \sqrt{r} \cdot r_Z = (1 - 2 \cdot \sqrt[4]{\delta'}) \cdot \sqrt{r} \cdot (1 - \sqrt[12]{\delta'}) \geq (1 - 3 \cdot \sqrt[12]{\delta'}) \cdot \sqrt{r}.$$
- Finally, we set $C' \stackrel{\text{def}}{=} (D')^2$. It is not hard to see that C' is a linear code over $\mathbb{F}$ with relative distance δ' and rate
$$\left[(1 - 3 \cdot \sqrt[12]{\delta'}) \cdot \sqrt{r}\right]^2 = \left(1 - 3 \cdot \sqrt[12]{\delta'}\right)^2 \cdot r \geq (1 - 6 \cdot \sqrt[12]{\delta'}) \cdot r.$$

Thus, C' has the required parameters. It remains to analyze the query complexity and running time of the local tester of C'. The basic idea of the proof is as follows: we observe that C' can be obtained from C by applying "local operations", and show that such local operations preserve the local testability. The rest of this section is organized as follows: in Section 5.2.1, we set up a framework for working with local operations. Then, in Section 5.2.2, we show that C' is locally testable by showing that it can be obtained from C using local operations, thus completing the proof of Lemma 5.6.

[6]The code meeting the the special case of the Zyablov bound was chosen just for concreteness of parameters. Any explicit family of binary linear codes with distance δ and rate $1 - \mathrm{poly}(\delta)$ (for arbitrary δ) would have sufficed for this construction.

5.2.1 Local operations

Block-wise operations. The first type of local operations that we use is block-wise operations, defined as follows.

Definition 5.7 (Block-wise operations)**.** Let $\phi : \Sigma^p \to \Sigma^m$ be one-to-one, and let $w \in \Sigma^n$ such that p divides n. We say that $w' \in \Sigma^{n'}$ is *obtained by applying ϕ to w block-wise* if it holds that

$$w' = \phi(w_1 \dots w_p)\, \phi(w_{p+1} \dots, w_{2p}) \cdots \phi(w_{n-p+1} \dots, w_n).$$

Observe that applying ϕ to w block-wise is a one-to-one function. For a set $W \subseteq \Sigma^n$, we say that a set $W' \subseteq \Sigma^{n'}$ is *obtained by applying ϕ to W block-wise* if W' is the result of applying ϕ block-wise to all the elements $w \in W$.

We say that ϕ is *invertible in time T* if there is an algorithm that takes as an input a string $u \in \Sigma^m$, runs in time T , and computes its pre-image $\phi^{-1}(u)$ (or rejects if it does not exist). We refer to the latter algorithm as the *inverter* of ϕ.

The following proposition shows that block-wise operations preserve local testability.

Proposition 5.8 (Local testability of block-wise operations)**.** *Let $L \subseteq \Sigma^n$ be a code that is locally testable with query complexity q, and let $\phi : \Sigma^p \to \Sigma^m$ be one-to-one. Then, the code $L' \subseteq \Sigma^{n'}$ that is obtained by applying ϕ to L block-wise is locally testable with query complexity $O(m^2 \cdot q)$.*

Furthermore, if ϕ is invertible in time T, then there is an oracle algorithm that computes the local tester of the corresponding code L' when given black box access to the local tester of L and to the inverter of ϕ. The resulting local tester of L' runs in time $O(m \cdot T \cdot T_L)$ where T_L is the running time of the local tester of L.

Proof. Let A be the local tester of L. We describe a local tester A' for L'. When given oracle access to a purported codeword $z' \in \Sigma^{n'}$, the local tester A' acts as follows. First, A' partitions z' to blocks of length m, chooses a uniformly distributed block, checks that this block is an image of ϕ, and rejects otherwise.

Next, A' emulates A. Whenever A makes a query to a coordinate $i \in [n]$, the tester A' acts as follows: A' reads the block of z' to which i belongs, i.e., the block whose index is $\lceil i/p \rceil$. If this block is not an image of ϕ, the tester A' rejects. If the block is an image of ϕ, the tester A' inverts it, retrieves the value of the i-th coordinate from the pre-image, and feeds it to A as an answer to the query. Finally, when A finishes running, A' accepts if A accepts and rejects otherwise.

It is easy to see that the query complexity of A' is $(q+1) \cdot m$, that it has the required running time, and satisfies the completeness property. We show that A' rejects z' with probability at least $\frac{1}{8 \cdot m} \cdot \mathrm{dist}(z', L')$ – this can be amplified to $\mathrm{dist}(z', L')/4$ at the cost of increasing the query complexity and running time by a factor of $O(m)$, which will give us the required result (see the discussion in Section 2.3).

Let y' be the string that is obtained by replacing each block of z' with the closest image of ϕ. Suppose first that $\mathrm{dist}(y', z') \geq \mathrm{dist}(z', L')/2$. In this case, at least $\mathrm{dist}(z', L')/2$ fraction of the blocks of z' are not images of ϕ, and hence A' rejects in the first step with probability at least $\mathrm{dist}(z', L')/2$.

Consider now the case where that $\mathrm{dist}(y', z') < \mathrm{dist}(z', L')/2$. In this case, the triangle inequality implies that $\mathrm{dist}(y', L') \geq \mathrm{dist}(z', L')/2$. Let y be the string obtained by inverting ϕ on all the blocks of y'. It is not hard to see that when A' emulates a query i of the tester A, it either answers it

with y_i or rejects. Hence, the rejection probability of A' on z' is at least the rejection probability of A on y, which is at least $\mathrm{dist}(y, L)/4$. Now, observe that

$$\mathrm{dist}(y', L') \cdot n' \leq m \cdot \mathrm{dist}(y, L) \cdot n,$$

and therefore

$$\mathrm{dist}(y, L) \geq \frac{n'}{n} \cdot \frac{\mathrm{dist}(y', L')}{m} \geq \frac{\mathrm{dist}(z', L')}{2 \cdot m}.$$

It thus follows that A' rejects z' with probability at least $\frac{1}{8 \cdot m} \cdot \mathrm{dist}(z', L')$, as required. ■

Permutations. The second type of local operations that we use is permuting the coordinates of a code:

Definition 5.9 (Permutations)**.** Let $\sigma : [n] \to [n]$ be a permutation, and let $w \in \Sigma^n$. We say that $w' \in \Sigma^n$ is *obtained by applying σ to w* if it holds that $w'_i = w_{\sigma(i)}$ for every $i \in [n]$. For a set $W \subseteq \Sigma^n$, we say that a set $W' \subseteq \Sigma^n$ is *obtained by applying σ to W* if W' is the result of applying σ to all the elements $w \in W$.

We say that σ is *invertible in time T* if there is an algorithm that takes as an input a coordinate $j \in [n]$, runs in time T, and computes its pre-image $\sigma^{-1}(j)$. We refer to the latter algorithm as the *inverter of σ*.

It is easy to see that applying a permutation preserves local testability. The following proposition states this fact along with the effect of the permutation on the running time of the tester.

Proposition 5.10 (Local testability of permutations)**.** *Let $L \subseteq \Sigma^n$ be a code that is locally testable with query complexity q, and let $\sigma : [n] \to [n]$ be a permutation. Then, the code L' that is obtained by applying σ to L is locally testable with query complexity q.*

Furthermore, if σ is invertible in time T, then there is an oracle algorithm that computes the local tester of the corresponding code L' when given black box access to the local tester of L and to the inverter of σ. The resulting local tester of L' runs in time $O(T \cdot T_L)$ where T_L is the running time of the local tester of L.

Local operations and tensor products. The following propositions describe the interaction between the above local operations and tensor products of codes. This will be useful for analyzing the local testability properties of the $\square$-distance amplification procedure, which applies local operations to a tensor product code.

Proposition 5.11. *Let $\mathbb{F}$ be a finite field, and let $\phi : \mathbb{F}^p \to \mathbb{F}^m$ be a linear one-to-one function. Let $L \subseteq \mathbb{F}^n$ be a linear code, and let $L' \subseteq \mathbb{F}^{n'}$ be the code that is obtained by applying ϕ to L block-wise. Then, there exist a linear one-to-one function $\phi' : \mathbb{F}^{p^2} \to \mathbb{F}^{m^2}$ and permutations σ_1, σ_2 such that $(L')^2$ is obtained by applying to L^2 the permutation σ_1, then ϕ' block-wise, and then the permutation σ_2.*

Furthermore, σ_1 and σ_2 are invertible in time $\mathrm{polylog}\, n'$*. The functions ϕ and ϕ' are invertible in time* $\mathrm{poly}(m)$ *since they are linear.*

Proof. Let G be a generating matrix of L, and let A be an $m \times p$ matrix such that $\phi(x) = A \cdot x$ for every $x \in \mathbb{F}^p$. Let $I_{n/p}$ be the $(n/p) \times (n/p)$ identity matrix. It is not hard to see that applying

ϕ block-wise to a vector $w \in \mathbb{F}^n$ is equivalent to multiplying w by $I_{n/p} \otimes A$. Hence, a generating matrix of L' is

$$(I_{n/p} \otimes A) \cdot G.$$

By Fact 5.1, the matrix $G \otimes G$ is a generating matrix of L^2, and a generating matrix of $(L')^2$ is

$$\big((I_{n/p} \otimes A) \cdot G\big) \otimes \big((I_{n/p} \otimes A) \cdot G\big).$$

By Fact 2.2, the latter matrix is equal to the matrix

$$\big(I_{n/p} \otimes A \otimes I_{n/p} \otimes A\big) \cdot (G \otimes G).$$

In other words, codewords of $(L')^2$ are obtained by multiplying codewords of L^2 by the matrix

$$I_{n/p} \otimes A \otimes I_{n/p} \otimes A.$$

Now, it is not hard to see[7] that by permuting the rows and columns of the latter matrix we can get the matrix

$$I_{n/p} \otimes I_{n/p} \otimes A \otimes A = I_{n^2/p^2} \otimes A \otimes A,$$

which is the matrix that applies the operation $A \otimes A$ block-wise. In other words, there exist permutation matrices P_1, P_2 such that

$$I_{n/p} \otimes A \otimes I_{n/p} \otimes A = P_2 \cdot \big(I_{n^2/p^2} \otimes A \otimes A\big) \cdot P_1,$$

and therefore a generating matrix of $(L')^2$ is

$$P_2 \cdot \big(I_{n^2/p^2} \otimes A \otimes A\big) \cdot P_1 \cdot (G \otimes G).$$

Let σ_1 and σ_2 be the permutations that correspond to the matrices P_1 and P_2. The latter expression means that $(L')^2$ is obtained by applying to L^2 the permutation σ_1, then $A \otimes A$ block-wise, then the permutation σ_2, as required. It is not hard to see that the permutations σ_1 and σ_2 are invertible in time $\operatorname{poly}\log(n)$. ■

Proposition 5.12. *Let $\mathbb{F}$ be a finite field, and let $\sigma : [n] \to [n]$ be a permutation. Let $L \subseteq \mathbb{F}^n$ be a linear code, and let $L' \subseteq \mathbb{F}^n$ be the code that is obtained by applying σ to L. Then, there exists a permutation σ' such that $(L')^2$ is obtained by applying σ' to L^2.*

Furthermore, if σ is invertible in time T then σ' is invertible in time $O(T)$.

The proof of Proposition 5.12 is similar to that of Proposition 5.11 but simpler, and is therefore omitted.

[7] To see it, let use denote $M \stackrel{\text{def}}{=} I_{n/p} \otimes A \otimes I_{n/p} \otimes A$ and $N \stackrel{\text{def}}{=} I_{n/p} \otimes I_{n/p} \otimes A \otimes A$. Since those matrices are tensor products, we can label the rows and the columns with quadruples of indices (i_1, i_2, i_3, i_4) and (j_1, j_2, j_3, j_4). Now, by the definition of the tensor product of matrices, it holds that

$$M_{(i_1,i_2,i_3,i_4),(j_1,j_2,j_3,j_4)} = \big(I_{n/p}\big)_{i_1,j_1} \cdot A_{i_2,j_2} \cdot \big(I_{n/p}\big)_{i_3,j_3} \cdot A_{i_4,j_4} = \big(I_{n/p}\big)_{i_1,j_1} \cdot \big(I_{n/p}\big)_{i_3,j_3} \cdot A_{i_2,j_2} \cdot A_{i_4,j_4} = N_{(i_1,i_3,i_2,i_4),(j_1,j_3,j_2,j_4)}.$$

Hence, N can be obtained from M by permuting the rows and the columns, as required.,

5.2.2 The local testability of C'

In this section, we show that the code C' is locally testable, thus completing the proof of Lemma 5.6. This is done in three steps: first, we observe that D' is obtained from D by applying a sequence of block-wise operations and permutations. We then use Propositions 5.11 and 5.12 to show that the same holds for C' and C. Finally, we use the local testability of C and Propositions 5.8 and 5.10 to conclude that C' is locally testable.

Obtaining D' from D. Recall that D' is obtained from D by applying the distance amplification of Lemma 4.1 to D to obtain a code $\widetilde{D}$, and then concatenating the resulting code $\widetilde{D}$ with an inner code $Z \subseteq \{0,1\}^m$.

The construction of $\widetilde{D}$ from D in Lemma 4.1 provides an $\mathbb{F}_2$-linear bijection from the code D to the code $\widetilde{D}$. We give a brief review this bijection, in order to argue that it is local. Let n denote the block length of D. Given a codeword $d \in D$, the bijection maps it to a codeword $\widetilde{d} \in \widetilde{D}$ as follows:

1. Choose a Reed-Solomon code of block length $s \stackrel{\text{def}}{=} \text{poly}(1/(\varepsilon \cdot \delta))$, rate $1-\varepsilon$, dimension b, and alphabet $\{0,1\}^r$ for $r \stackrel{\text{def}}{=} \log s$.
2. The codeword d is partitioned to $n' \stackrel{\text{def}}{=} \frac{n}{b \cdot r}$ blocks of length $b \cdot r$.
3. Each block is viewed as a string of length b over the alphabet $\{0,1\}^r$, and is encoded with the Reed-Solomon code.
4. Together, the encoded blocks form a string w of length $n' \cdot s$ over the alphabet $\{0,1\}^r$. The coordinates of w are now permuted according to some fixed permutation, which is determined by the edges of a strongly-explicit expander. Let us denote the obtained string w'.
5. Finally, the string w' is partitioned to n' blocks of length s. Let us denote those blocks by $S_1, \ldots, S_{n'}$. We view each block S_j as a single symbol over the alphabet $\Sigma = \{0,1\}^{r \cdot s}$, and define $\widetilde{d}$ to be the resulting string over Σ.

It is easy to see that the code D' is obtained from D by applying the same bijection, and then encoding the blocks $S_1, \ldots, S_{n'}$ with the inner code Z rather than viewing them as symbols. Now, observe that this corresponds to obtaining D' from D by applying a block-wise operation (the encoding with Reed-Solomon), a permutation, and then another block-wise operation (the encoding with Z).

Obtaining C' from C. Let D_1 be the code obtained from D after applying the first block-wise operation, and let D_2 be the code obtained from D_1 by applying the permutation. Observe that D' is obtained by applying a block-wise operation to D_2.

Let $C_1 = (D_1)^2$ and $C_2 = (D_2)^2$. By Propositions 5.11 and 5.12, the following claims hold:

1. C_1 is obtained by applying to $C = D^2$ a permutation, a block-wise operation, and then another permutation. Specifically, the block-wise operation has block length $(s \cdot r)^2 \leq \text{poly}(1/\cdot\,\delta)$.
2. C_2 is obtained by applying a permutation to C_1.

3. $C' = (D')^2$ is obtained by applying to C_2 a permutation, a block-wise operation, and then another permutation. Specifically, the block-wise operation has block length m^2 (where m is the block length of Z).

The local testability of C'. We conclude that C' is obtained from C by applying permutations and two block-wise operations. The blocks of the block-wise operations are of length $\mathrm{poly}(m/(\varepsilon \cdot \delta))$ (where m is the block length of Z). By Propositions 5.8 and 5.10, it follows that C' is locally testable with query complexity $q \cdot \mathrm{poly}(m/\delta) = q \cdot \mathrm{poly}(1/\delta)$.

We turn to analyze the running time of the local tester. Observe that the block-wise operations are invertible in time $\mathrm{poly}(m/\delta)$: the reason is that those operations are linear, and generating matrices of the Reed-Solomon code and the Zyablov code Z can be constructed in time $\mathrm{poly}(s) \leq \mathrm{poly}(1/\delta)$ and $\mathrm{poly}(m)$ respectively.

Next, observe that the permutations are invertible in time $\mathrm{poly}(m \cdot \log n/\delta)$: For the permutations that obtain C_1 from C and C' from C_2, this follows from Proposition 5.11. For the permutation that obtains C_2 from C_1, this follows from Proposition 5.12, and from the fact that the above bijection determines the permutation according to a strongly-explicit expander.

Propositions 5.8 and 5.10 imply in turn that the local tester of C' runs in time

$$\mathrm{poly}(m \cdot \log n/\delta) \cdot T_C = \mathrm{poly}(\log n/\delta) \cdot T_C$$

as required.

5.3 Proof of Lemma 4.2

We construct the code W_n as follows: We start with a code of block length $\mathrm{poly}\log(n)$, rate $1 - \frac{1}{\mathrm{poly}\log(n)}$, and relative distance $\frac{1}{\mathrm{poly}\log(n)}$. This code is trivially locally testable using $\mathrm{poly}\log(n)$ queries – one can simply read the entire codeword. Then, we iteratively increase the block length of the code using the tensor product operation, while using the $\square$-distance amplification to maintain the relative distance. The query complexity increases by a factor of $\mathrm{poly}\log(n)$ in each iteration, and the rate is roughy squared (actually, it is squared and then multiplied by $1 - \frac{1}{\mathrm{poly}\log(n)}$). Hence, after about $\log\log n$ iterations, we end up with a code of block length n, rate $1 - \frac{1}{\mathrm{poly}\log(n)}$, relative distance $\frac{1}{\mathrm{poly}\log(n)}$, and query complexity $(\log n)^{O(\log\log n)}$ as required. Details follow.

The construction. In what follows, we assume that the block length n is sufficiently large to make all the inequalities hold – otherwise, n is smaller than some large constant, so we can choose W_n to be any code, and it will be locally testable with a constant number of queries.

Let $n \in \mathbb{N}$. The code $W \stackrel{\mathrm{def}}{=} W_n$ is constructed as follows. We construct a sequence of binary linear codes $B_0, B_1, \ldots$ and set $W = B_t$ for sufficiently large t to be determined later on. All the codes in the sequence will have relative distance at least $\delta \stackrel{\mathrm{def}}{=} \frac{1}{(\log n)^{36}}$. We choose B_0 to be the Zyablov code[8] of Fact 2.7 with relative distance δ and rate at least $1 - \sqrt[3]{\delta}$, and choose its block length to be minimal such that a code with those parameters exists – this block length can be $\mathrm{poly}\log(n)$. For every $i \geq 1$, we choose B_i to be the result of applying the $\square$-distance amplification (Lemma 5.6) to $(B_{i-1})^2$ to amplify the relative distance from δ^2 to δ.

[8] As before, the choice of Zyablov code is not crucial, and is chosen for concreteness of parameters.

Let us denote by n_0 the block length of B_0. Clearly, the block length of B_i is $(n_0)^{2^i}$. We therefore set W to be B_t for

$$t \stackrel{\text{def}}{=} \log_2 \log_{n_0} n \leq \log \log n,$$

so the block length of W is n. It is also clear that all the codes B_i have relative distance at least δ, so in particular W has relative distance has at least $\delta = \frac{1}{\text{poly}\log(n)}$, as required. It can also be seen that the family $\{W_n\}_n$ is explicit and linear. It remains to analyze the rate and the query complexity of W, and the running time of the tester.

The rate. In order to lower bound the rate of W, we prove the following claim, which gives a lower bound on the rate of each B_i.

Claim 5.13. *The rate of B_i is at least $(1 - \frac{6}{\log^3 n})^{3^i}$.*

Proof. We prove the claim by induction. For $i = 0$, the required claim holds by the definition of B_0. Suppose $i \geq 1$. Recall that the code B_i was obtained by applying Lemma 5.6 to $(B_{i-1})^2$ to amplify the relative distance to δ. By the induction assumption, the rate of B_{i-1} is at least $r_{i-1} \stackrel{\text{def}}{=} (1 - \frac{6}{\log^3 n})^{3^{i-1}}$. Thus, the rate of $(B_{i-1})^2$ is at least $(r_{i-1})^2$, and by Lemma 5.6, the rate of B_i is at least

$$\begin{aligned}
(1 - 6 \cdot \sqrt[12]{\delta}) \cdot r_{i-1}^2 &= (1 - \frac{6}{\log^3 n}) \cdot (1 - \frac{6}{\log^3 n})^{3^{i-1} \cdot 2} \\
&= (1 - \frac{6}{\log^3 n})^{3^{i-1} \cdot 2 + 1} \\
&\geq (1 - \frac{6}{\log^3 n})^{3^i},
\end{aligned}$$

as required. ■

The last claim implies in particular that the rate of W is at least

$$\begin{aligned}
(1 - \frac{6}{\log^3 n})^{3^t} &\geq 1 - \frac{6}{\log^3 n} \cdot 3^t \\
&\geq 1 - \frac{6}{\log^3 n} \cdot 3^{\log \log n} \\
&\geq 1 - \frac{6}{\log^3 n} \cdot \log^2 n \\
&\geq 1 - \frac{6}{\log n},
\end{aligned}$$

as required.

The query complexity. We turn to analyze the query complexity of W. The running time of the tester can be analyzed similarly. We prove the following claim, which gives an upper bound on the query complexity of each B_i.

Claim 5.14. *There exists constants $c_0, c_1 \in \mathbb{N}$ such that the query complexity of B_i is at most $(\log n)^{c_1 \cdot i + c_0}$.*

Proof. We prove the claim by induction. Recall that n_0 is that block length of B_0. We choose c_0 to be such that $n_0 \leq \log^{c_0} n$, so the base case of the induction holds. We turn to prove the claim for $i \geq 1$.

Recall that the code B_i is obtained by applying Lemma 5.6 to $(B_{i-1})^2$. By the induction assumption, the query complexity of B_{i-1} is at most $q_{i-1} = (\log n)^{c_1 \cdot (i-1) + c_0}$. By Corollary 5.5, the query complexity of $(B_{i-1})^2$ is at most

$$q'_{i-1} \stackrel{\text{def}}{=} 1200 \cdot q_{i-1}/\delta^6 \leq q_{i-1}/\delta^7.$$

By Lemma 5.6, there exists some constant c' such that the query complexity of B_i is at most

$$\begin{aligned} q'_{i-1}/\delta^{c'} &\leq q_{i-1}/\delta^{c'+7} \\ &\leq q_{i-1} \cdot (\log n)^{24 \cdot (c'+7)} \\ &\leq (\log n)^{c_1 \cdot (i-1) + c_0 + 24 \cdot (c'+7)}. \end{aligned}$$

Now, by setting $c_1 \stackrel{\text{def}}{=} 24 \cdot (c' + 7)$, we get that the query complexity of B_i is at most $(\log n)^{c_1 \cdot i + c_0}$, as required. ■

The last claim implies in particular that the query complexity of W is at most

$$(\log n)^{c_1 \cdot t + c_0} \leq (\log n)^{c_1 \cdot \log \log n + c_0} = (\log n)^{O(\log \log n)},$$

as required.

Acknowledgement. We would like to thank Irit Dinur, Tali Kaufman 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 a preliminary version of this work.

References

[ABN+92] Noga Alon, Jehoshua Bruck, Joseph Naor, Moni Naor, and Ron M. Roth. Construction of asymptotically good low rate error-correcting codes through pseudo-random graphs. *IEEE Transactions on Information Theory*, 38:509–516, 1992.

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

[AEL95] Noga Alon, Jeff Edmonds, and Michael Luby. Linear time erasure codes with nearly optimal recovery. In *proceedings of the 36th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 512–519. IEEE Computer Society, 1995.

[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.

[AM85] N. Alon and V. D. Milman. λ_1, Isoperimetric 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*, 45(1):70–122, 1998.

[BFLS91] László Babai, Lance Fortnow, Leonid A. Levin, and Mario Szegedy. Checking computations in polylogarithmic time. In *Proceedings of the 23rd Annual ACM Symposium on Theory of Computing (STOC)*, pages 21–31. ACM Press, 1991.

[BFNW93] László Babai, Lance Fortnow, Noam Nisan, and Avi Wigderson. BPP has subexponential time simulations unless EXPTIME has publishable proofs. *Computational Complexity*, 3(4):307–318, 1993.

[BK95] Manuel Blum and Sampath Kannan. Designing programs that check their work. *Journal of ACM*, 42(1):269–291, 1995.

[BLR93] Manuel Blum, Michael Luby, and Ronitt Rubinfeld. Self-testing/correcting with applications to numerical problems. *J. Comput. Syst. Sci.*, 47(3):549–595, 1993.

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

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

[BV09] Eli Ben-Sasson and Michael Viderman. Tensor products of weakly smooth codes are robust. *Theory of Computing*, 5(1):239–255, 2009.

[BV12] Eli Ben-Sasson and Michael Viderman. Towards lower bounds on locally testable codes via density arguments. *Computational Complexity*, 21(2):267–309, 2012.

[BV15] Eli Ben-Sasson and Michael Viderman. Composition of semi-LTCs by two-wise tensor products. *Computational Complexity*, 24(3):601–643, 2015.

[CKGS98] Benny Chor, Eyal Kushilevitz, Oded Goldreich, and Madhu Sudan. Private information retrieval. *J. ACM*, 45(6):965–981, 1998.

[CR05] Don Coppersmith and Atri Rudra. On the robust testability of tensor products of codes. *Electronic Colloquium on Computational Complexity (ECCC)*, (104), 2005.

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

[DK11] Irit Dinur and Tali Kaufman. Dense locally testable codes cannot have constant rate and distance. In *proceedings of the 14th International Workshop on Randomization and Computation (RANDOM)*, pages 507–518. Springer, 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 *proceedings of the 9th International Workshop on Randomization and Computation (RANDOM)*, pages 304–315. Springer, 2006.

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

[For66] David Forney. Generalized minimum distance decoding. *IEEE Transactions on Information Theory*, 12(2):125–131, 1966.

[FS95] Katalin Friedl and Madhu Sudan. Some improvements to total degree tests. In *proceedings of the 3rd Israel Symposium on the Theory of Computing and Systems (ISTCS)*, pages 190–198. IEEE Computer Society, 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 *Proceedings of the 34th Annual ACM Symposium on Theory of Computing (STOC)*, pages 812–821. ACM Press, 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 *proceedings of the Innovations in Theoretical Computer Science Conference (ITCS)*, pages 529–540. ACM Press, 2013.

[GM12] Oded Goldreich and Or Meir. The tensor product of two good codes is not necessarily locally testable. *Inf. Proces. Lett.*, 112(8-9):351–355, 2012.

[Gol11] Oded Goldreich. Bravely, moderately: a common theme in four recent works. In *Studies in Complexity and Cryptography. Miscellanea on the Interplay between Randomness and Computation*, pages 373–389. Springer, 2011.

[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.

[GS06] Oded Goldreich and Madhu Sudan. Locally testable codes and PCPs of almost linear length. *Journal of ACM*, 53(4):558–655, 2006.

[Ham50] Richard W. Hamming. Error detecting and error correcting codes. *Bell System technical journal*, 29(2):147–160, 1950.

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

[HOW15] Brett Hemenway, Rafail Ostrovsky, and Mary Wootters. Local correctability of expander codes. *Inf. Comput*, 243:178–190, 2015.

[HW15] Brett Hemenway and Mary Wootters. Linear-time list recovery of high-rate expander codes. In *Proceedings of the 42nd International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 701–712. Springer, 2015.

[KMRS15a] Swastik Kopparty, Or Meir, Noga Ron-Zewi, and Shubhangi Saraf. High-rate locally-correctable and locally-testable codes with sub-polynomial query complexity. *Electronic Colloquium on Computational Complexity (ECCC)*, 2015.

[KMRS15b] Swastik Kopparty, Or Meir, Noga Ron-Zewi, and Shubhangi Saraf. High-rate locally-testable codes with quasi-polylogarithmic query complexity. *Electronic Colloquium on Computational Complexity (ECCC)*, 2015.

[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 *Proceedings of the 32nd Annual ACM Symposium on Theory of Computing (STOC)*, pages 80–86. ACM Press, 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.

[Lip90] Richard J. Lipton. Efficient checking of computations. In *proceedings of the 7th Annual ACM Symposium on Theoretical Aspects of Computer Science (STACS)*, volume 415 of *lncs*, pages 207–215. Springer, 1990.

[Mei09] Or Meir. Combinatorial construction of locally testable codes. *SIAM J. Comput.*, 39(2):491–544, 2009.

[Mei12] Or Meir. On the rectangle method in proofs of robustness of tensor products. *Inf. Process. Lett.*, 112(6):257–260, 2012.

[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.

[Rei08] Omer Reingold. Undirected connectivity in log-space. *J. ACM*, 55(4), 2008.

[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.

[RVW02] Omer Reingold, Salil Vadhan, and Avi Wigderson. Entropy waves, the zig-zag graph product, and new constant-degree expanders. *Annals of Mathematics*, 155(1):157–187, 2002.

[Sha48] Claude E. Shannon. A mathematical theory of communication. *ACM SIGMOBILE Mobile Computing and Communications Review*, 5(1):3–55, 2001. Originally appeared in *Bell System Tech. J.* 27:379–423, 623–656, 1948.

[Sin64] Richard C. Singleton. Maximum distance q -nary codes. *IEEE Transactions on Information Theory*, 10(2):116–118, 1964.

[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.

[Tre03] Luca Trevisan. List-decoding using the XOR lemma. In *proceedings of the 44th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 126–135. IEEE Computer Society, 2003.

[Val05] Paul Valiant. The tensor product of two codes is not necessarily robustly testable. In *proceedings of the 9th International Workshop on Randomization and Computation (RANDOM)*, pages 472–481. Springer, 2005.

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

[Vid12] Michael Viderman. Strong LTCs with inverse polylogarithmic rate and soundness. *Electronic Colloquium on Computational Complexity (ECCC)*, 19:168, 2012.

[Vid13] Michael Viderman. Strong LTCs with inverse poly-log rate and constant soundness. In *proceedings of the 54th IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 330–339. IEEE Computer Society, 2013.

[Vid15] Michael Viderman. A combination of testability and decodability by tensor products. *Random Struct. Algorithms*, 46(3):572–598, 2015.

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

[WW05] Stephanie Wehner and Ronald de Wolf. Improved lower bounds for locally decodable codes and private information retrieval. In *proceedings of the 32nd International Colloquium on Automata, Languages and Programming (ICALP)*, pages 1424–1436. Springer, 2005.

[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.

[Zya71] Victor V. Zyablov. An estimate on the complexity of constructing binary linear cascade codes. *Problems of Information Transmission*, 7(1):3–10, 1971.