High-rate Locally-testable Codes with Quasi-polylogarithmic Query Complexity

Swastik Kopparty[*], Or Meir[†], Noga Ron-Zewi[‡], Shubhangi Saraf[§]

September 5, 2015

Abstract

An error correcting code is said to be *locally testable* if there is a test that checks whether a given string is a codeword, or rather far from the code, by reading only a small number of symbols of the string. Locally testable codes (LTCs) are both interesting in their own right, and have important applications in complexity theory.

A long line of research tries to determine the best tradeoff between rate and distance that LTCs can achieve. In this work, we construct LTCs that have high rate (arbitrarily close to 1), have constant relative distance, and can be tested using $(\log n)^{O(\log \log n)}$ queries. This improves over the previous best construction of LTCs with high rate, by the same authors, which uses $\exp(\sqrt{\log n \cdot \log \log n})$ queries [KMRS15].

In fact, as in [KMRS15], our result is actually stronger: for binary codes, we obtain LTCs that match the Zyablov bound for any rate $0 < r < 1$. For codes over large alphabet (of constant size), we obtain LTCs that approach the Singleton bound, for any rate $0 < r < 1$.

1 Introduction

Locally-testable codes (LTCs) are error-correcting codes that admit local error-detecting algorithms. Specifically, LTCs are codes which come equipped with a randomized sub-linear time "local-testing algorithm", which when given oracle access to a received word z, makes a few queries to z and distinguishes between the following two cases with high probability:

- z is a codeword,
- z is ε-far from every codeword of the code.

LTCs were introduced by [FS95, RS96] and their systematic study was begun in [GS06]. They are not only interesting in their own right, but also have important connections to other areas of complexity theory, most notably to PCPs [AS98, ALM$^+$98].

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

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

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

[§]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

The three key parameters of LTCs are: the *rate*, the *relative distance*, and the *query complexity*. The *rate* of a code is the ratio of the message length to the codeword length, and it measures the amount of redundancy of the code. The *relative distance* of a code is the minimum fraction of coordinates on which every pair of codewords from the code disagree, and is related to the fraction of errors that the code can detect and correct. Finally, the *query complexity* of an LTC is the number of queries made by the local-testing algorithm to the received word. Naturally, one would like to maximize the rate and relative distance of the LTC while minimizing its query complexity. In particular, the most interesting and fundamental question about LTCs asks: are there LTCs with constant rate, constant relative distance and constant query complexity.

Most of the research on LTCs so far has focused on the constant query regime. After many years of intensive research [HS00, GS06, BSVW03, BGH$^+$06], it is now known [BS08, Din07, Vid15b] that there are LTCs of length n with rate $\Omega(1/\mathsf{polylog}(n))$, constant relative distance and constant query complexity (where the query complexity can even be as small as 3!). It is not known if one can obtain codes with constant rate and constant relative distance in this regime of constant query complexity, and it is known that if the LTCs have additional restrictions then constant rate and constant relative distance are not possible [DK11, BV12].

This paper is about the regime of LTCs with constant rate. In this regime one fixes the rate r and the relative distance to be constants and asks for the minimum query complexity achievable by LTCs. For constant rate $r < 1/2$, Reed-Muller codes over large fields were long known [RS96] to give LTCs of length n with rate r, constant relative distance and query complexity $\approx n^{O(1/(\log(1/r))}$. More recent work [Vid15a, GKS13] showed that in fact one can get LTCs with constant relative distance and query complexity $O(n^\beta)$ for any constant $\beta > 0$, even with rate arbitrarily close to 1. Even more recently, we showed in [KMRS15] that there exist LTCs with constant rate (and again the rate can be taken arbitrarily close to 1), constant relative distance, and query complexity $\exp(\sqrt{\log n \cdot \log\log n})$.

1.1 Our results

In this work, we construct LTCs that have constant rate (which can be taken arbitrarily close to 1) and constant relative distance, with query complexity being only $(\log n)^{O(\log\log n)}$. Formally, we prove the following result.

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

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

In fact, not only do our LTCs have constant rate and constant relative distance, but the trade-off between the rate and the distance matches *the Zyablov bound* [Zya71]. Formally, the relative distance δ above satisfies

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

where H is the binary entropy function.

Over large (but still constant size) alphabets, we show that such codes can approach the Singleton bound, i.e they can basically obtain a rate-distance tradeoff that is the best possible for general codes.

Theorem 1.2 (LTCs with quasi-polylogarithmic query complexity approaching the Singleton bound)**.** *For every $r \in (0,1)$ and every $\varepsilon > 0$, there exists an explicit infinite family of linear codes $\{C_n\}_n$ satisfying:*

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

We note that the exponential dependence of the alphabet size on ε follows from our use of the Alon-Luby distance-amplification method (see below). This dependence indeed seems to be a bottleneck in all applications of this method, e.g. [GI05].

1.2 Our techniques

Our construction of LTCs uses ingredients from our previous construction of high-rate LTCs [KMRS15], in conjunction with ideas from the iterative construction of constant-query LTCs by Meir [Mei09].

In [KMRS15], we showed that a technique of Alon and Luby [AL96] can be used to amplify the relative distance of LTCs while preserving their local testability. In particular, starting with an LTC with relative distance δ, we can amplify its relative distance to any $0 < \delta' < 1$ while increasing the query complexity by a factor of $\mathrm{poly}(1/\delta)$ and decreasing the rate by a factor of only $\approx 1 - \delta'$. This technique is useful, since it is easier to construct LTCs with small relative distance, and this technique allows us to transform such LTCs into ones with better relative distance. Indeed, our construction of LTCs in [KMRS15] followed this scheme: first, we constructed LTCs with constant rate and extremely small relative distance, namely $\exp(-\sqrt{\log n \cdot \log\log n})$. Then, we applied the Alon-Luby technique to amplify the relative distance to a constant, while paying a factor of $\exp(\sqrt{\log n \cdot \log\log n})$ in the query complexity.

The main technical contribution of this paper is an improved construction of LTCs in the sub-constant relative distance regime. More specifically, we show how to construct LTCs of high rate with relative distance $1/\mathsf{polylog}(n)$ and query complexity $(\log n)^{O(\log\log n)}$ (so both relative distance and query complexity are improved). Using the Alon-Luby distance-amplification, this gives in turn LTCs of high rate with constant relative distance and query complexity $(\log n)^{O(\log\log n)}$.

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 [RVW00], 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 Alon-Luby distance-amplification technique (but in a different setting of parameters).

1.2.1 The iterative construction

We now describe our iterative construction of high rate LTCs with relative distance $1/\mathsf{polylog}(n)$ and query complexity $(\log n)^{O(\log\log n)}$. 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 $\mathrm{poly}\log n$, rate $1 - \frac{1}{\mathrm{poly}\log n}$, and relative distance $\frac{1}{\mathrm{poly}\log n}$. 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 Alon-Luby 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, BV09b, BV09a, Vid15a].

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, Vid15a] 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 Alon-Luby 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 $\operatorname{poly}(1/\delta)$. Following [Mei09], we show it by decomposing this operation into simpler block-wise operations, and showing that each of these operations preserves local testability.

1.3 Related work

1.3.1 Comparison with [KMRS15]

As explained above, in our earlier work [KMRS15], we gave a construction of LTCs with constant rate, constant relative distance, and query complexity $\exp(\sqrt{\log n \cdot \log\log n})$. This construction had two parts: (1) the construction of an LTC with sub-constant relative distance, and (2) application of the Alon-Luby distance-amplification to this LTC to get constant relative distance (while roughly preserving the other parameters). Our technical contribution is improving the LTC that is constructed in the first step.

In order to make the comparison between the constructions easier, the first step of [KMRS15] could be presented as follows: As in the current construction, we start with a code with small block length, which is trivially locally testable, and then iteratively increase its block length. However, unlike our current construction, the iterations of [KMRS15] consist only of the tensor product operation, without using the distance amplification.

Since we do not amplify the distance in each iteration, it decays quickly, and we end up with relative distance $\exp(-\sqrt{\log n \cdot \log\log n})$. Moreover, since the tensor product increases the query complexity by a factor that depends on the relative distance, the query complexity increases faster, and we can afford less iterations. Hence, we end up with worse parameters compared to the current construction.

1.3.2 Comparison with [Mei09]

As mentioned above, our construction bears resemblance to the LTCs of [Mei09]. The main part of [Mei09] constructs LTCs with rate $\frac{1}{\operatorname{poly}\log(n)}$, constant relative distance, and query complexity[1] $\operatorname{poly}\log(n)$. That construction, too, starts with a code of small block length, which is trivially locally testable, and increases it iteratively. In each iteration, the block length is squared, the rate decreases by a constant factor, the relative distance is maintained, and the query complexity increases by a constant factor. Hence, after $O(\log\log n)$ iterations, we get the required parameters.

A single iteration of [Mei09] consists, too, of applying tensor product and distance amplification to the code. However, a single iteration there also applies an additional operation called "random projection", which partially undoes the rate loss causes by tensoring - it causes the rate to decrease by a constant factor rather than be squared in each iteration.

The difference between the two constructions could therefore be summed up as follows:

- The construction of [Mei09] starts with a constant rate, and then decreases the rate by a constant factor in each iteration. Thus, after $O(\log\log n)$ iterations, it ends up with rate $\frac{1}{\operatorname{poly}\log(n)}$.
- Our construction starts with rate $1 - \frac{1}{\operatorname{poly}\log n}$, and then squares the rate in each iteration. Thus, after $O(\log\log n)$ iterations, it ends up with constant rate. A crucial point here is that when the rate is very high (e.g., $1 - \frac{1}{\operatorname{poly}\log n}$), squaring the rate is better than multiplying the rate by a constant factor.

The difference between the constructions may seem as only a matter of choosing the parameters, which raises the question why [Mei09] could not obtain our construction. There are two reasons for that:

[1] The query complexity is decreased to a constant in a later part of [Mei09].

- The construction of [Mei09] uses a different distance-amplification method due to [ABN+92]. This method always loses at least a constant factor in the rate, regardless of the choice of parameters. Hence, [Mei09] could not avoid a loss of constant factor in each iteration, and had to end up with rate $\frac{1}{\text{poly}\log(n)}$. On the other hand, our use of the Alon-Luby amplification method allows us to lose only a factor of $1 - \frac{1}{\text{poly}\log n}$ in each iteration.
- In analyzing the local testability of the tensor product, [Mei09] relied on a theorem of [BS06]. This theorem only holds for codes that have very high relative distance ($\approx \sqrt[4]{\frac{7}{8}}$). Thus, [Mei09] had to work with codes with very high relative distance, which forced those codes to have low rate. In particular, [Mei09] could not start with rate $1 - \frac{1}{\text{poly}\log n}$.
  We, on the other hand, replace the theorem of [BS06] with a more recent theorem of [Vid15a], which also works for codes of low distance. Hence, we can work with codes of rate $1 - \frac{1}{\text{poly}\log n}$.

An interesting research direction would be to 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.

1.4 Organization

We review preliminaries regarding error-correcting codes and locally-testable codes in Section 2. In Sections 3 and 4 we formally define the tensor product and $\square$-distance amplification operations respectively, and analyze the effects these operations have on the parameters of LTCs. Finally, in Section 5, we construct our LTCs using these operations.

2 Preliminaries

All logarithms in this paper are in base 2 unless specified otherwise. 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 $\text{dist}(x, y) \stackrel{\text{def}}{=} |\{i \in [n] : x_i \neq y_i\}| / n$. We have the following useful approximation.

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

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

Proof. It holds that

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

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 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 $\Sigma = \mathbb{F}$, then 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*. We say that C has *relative distance* at least δ if for every pair of distinct codewords $c_1, c_2 \in C$ it holds that $\mathrm{dist}(c_1, c_2) \geq \delta$. We 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 to* (respectively, ε-*far from*) C if $\mathrm{dist}(z, C) < \varepsilon$ (respectively, if $\mathrm{dist}(z, C) \geq \varepsilon$).

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 isomorphism from $\mathbb{F}^k$ to C. We say that an infinite family of linear codes $\{C_n\}_n$ is *explicit* if there is an algorithm that on input n outputs a generating matrix of C_n in time $\mathrm{poly}(n)$.

Zyablov codes. We use the following fact, which states the existence of the Zyablov codes.

Fact 2.3 (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.

In the latter statement of the Zyablov bound, we chose δ as a function of r. However, we may also choose r as a function of δ, which leads to the following statement: for every $0 < \delta < 1$ and $\varepsilon > 0$, there exists an explicit infinite family $\{Z_n\}_n$ of binary linear codes of with relative distance δ and rate

$$r = \max_{0 < R < 1 - H(\delta + \varepsilon)} \left\{ R \cdot \left(1 - \frac{\delta}{H^{-1}(1 - R) - \varepsilon}\right) \right\},$$

where H is the binary entropy function. In particular, for small values of δ, we can choose $\varepsilon = \sqrt{\delta}$ and $R = 1 - H(2 \cdot \sqrt{\delta})$, yielding the following result.

Fact 2.4 (Special case of the Zyablov bound)**.** *For every sufficiently small $\delta > 0$, there exists an explicit infinite family $\{Z_n\}_n$ of binary linear codes of with relative distance δ and rate at least*

$$r = \left(1 - H(2 \cdot \sqrt{\delta})\right) \cdot \left(1 - \sqrt{\delta}\right) \geq 1 - \sqrt[3]{\delta},$$

where the inequality holds for sufficiently small values of δ.

2.2 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. From now on, we will work exclusively with strong LTCs, since it is a simpler notion and allows us to state a stronger result.

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

- **Input:** A gets oracle access to a string $z \in \Sigma^n$.
- **Completeness:** If z is a codeword of C, then A accepts with probability 1.
- **Soundness:** If z is not a codeword of C, then A rejects with probability at least $\mathrm{dist}(z, C)/4$.
- **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. We will often also be interested in the running time of the uniform local tester.

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

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

Our definition corresponds to the special case where $\rho = \frac{1}{4}$. 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 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 amplified tester rejects z 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.

3 Tensor product

In this section, we provide the formal definition of the tensor product operation and state the effect of this operation on the parameters of the LTC. The effect of this operation on the classical parameters of a code such as the block length, rate and relative distance is well-known (see, e.g.

[Sud01, DSW06]). To argue about the effect of this operation on the query complexity of an LTC we shall use a result due to Viderman [Vid15a].

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 3.1 (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 $G_1 \otimes G_2$).*

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-dimnesional 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 3.2 (see, e.g., [Mei09, Section 4.1])**.** *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 3.2 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, Vid15a] for the purpose of constructing locally testable codes using tensor products. In particular, we use the following result.

Theorem 3.3 ([Vid15a, 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,*

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

Following [Mei09], we use the latter theorem for showing that the tensor product operation maintains the local testability of codes:

Corollary 3.4 (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 $\left(O(T) + \operatorname{poly}\log(n)\right)/\delta^6$.*

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 2.2).

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 3.3, 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. ■

4 □-distance amplification

In this section, we describe the □-distance amplification operation and analyze its effect on the parameters of an LTC. As explained in the introduction, the □-distance amplification operation 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 Alon-Luby 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 one more issue that needs to be handled: the Alon-Luby 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 Alon-Luby amplification, we concatenate the resulting code with a binary inner code to reduce the alphabet size back to 2.

We turn to implementing the above ideas formally. We start by describing the two main ingredients of the □-distance amplification operation: the Alon-Luby distance-amplification and concatenation.

Alon-Luby distance-amplification. Recall that that in our previous work [KMRS15] we observed that a technique of Alon and Luby [AL96] can be used to amplify the relative distance of LTCs. We now state the lemma that we need from [KMRS15] (actually, a special case of this lemma for binary linear codes).

Lemma 4.1 (Alon-Luby distance-amplification, [KMRS15, Lemma 4.2])**.** *Suppose that there exists a binary linear code $C \subseteq \{0,1\}^n$ with relative distance δ and rate r that is locally testable with 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 $q \cdot \mathrm{poly}(1/(\varepsilon \cdot \delta))$ such that:*

- *$|C'| = |C|$.*
- *C' has rate at least $r \cdot (1 - \delta' - \varepsilon)$.*
- *The alphabet of C' is $\Sigma \stackrel{\text{def}}{=} \{0,1\}^p$ for some $p = \mathrm{poly}(1/(\varepsilon \cdot \delta))$.*
- *C' is $\mathbb{F}_2$-linear.*

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 ε.*
- *There is an oracle algorithm that computes the local tester of the code C' when given black box access to the local tester of the code C, and given also r, δ, δ', ε, and the block length of C. Moreover, if the local tester of C runs in time T_C, then the resulting local tester of C' runs in time $T_C \cdot \mathrm{poly}\left(\log(n)/(\varepsilon \cdot \delta)\right)$, where n is the block length of C.*

Remark 4.2. The running time that is stated in Lemma 4.1 for the tester of C' is better than the one that is stated in [KMRS15, Lemma 4.2]. However, the better running time can be verified by inspecting the proof in [KMRS15].

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 such that $\Sigma = \Lambda^p$ for some $p \in \mathbb{N}$. Let $C \subseteq \Sigma^n$ be a code over Σ and let $H \subseteq \Lambda^m$ be a code over Λ. Suppose there exists a bijection $\phi : \Lambda^p \to H$. 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 4.3 (Concatenation)**.** *Let $C \subseteq \Sigma^n$ be a code over $\Sigma = \Lambda^p$ with rate r_C and relative distance δ_C, let $H \subseteq \Lambda^m$ be a code over Λ 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.*

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

Lemma 4.4 (Properties of $\square$-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 $q \cdot \mathrm{poly}(1/\delta)$ 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 C.*
- *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 Alon-Luby 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 2.4 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 Alon-Luby 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 2.4)[2] 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 4.1, we set up a framework for working with local operations, which is a special case of framework of [Mei09]. Then, in Section 4.2, 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 4.4.

[2]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.

4.1 Local operations

4.1.1 Block-wise operations

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

Definition 4.5 (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 (this is a variant of Corollary 5.18 in [Mei09]).

Proposition 4.6 (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 2.2).

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 $\mathrm{dist}(z', L')/8$.

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. ■

4.1.2 Permutations

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

Definition 4.7 (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 4.8 (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.

4.1.3 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 4.9 (Follows from [Mei09, Propositions 5.5,5.6])**.** *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 $\mathsf{polylog} n'$*. 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 3.1, 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[3]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 4.10. *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 4.10 is similar to that of Proposition 4.9 but simpler, and is therefore omitted.

4.2 The local testability of C'

In this section, we show that the code C' is locally testable, thus completing the proof of Lemma 4.4. 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 4.9 and 4.10 to show that the same holds for C' and C. Finally, we use the local testability of C and Propositions 4.6 and 4.8 to conclude that C' is locally testable.

[3]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.,

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 description of this bijection, in order to argue that it is local (see [KMRS15] for more details). 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[4] 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 4.9 and 4.10, 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).

[4] For simplicity, we assume here that $b \cdot r$ divides n. In the general case, we increase n to the next multiple of $b \cdot r$ by padding d with zeroes (see [KMRS15, Section 3.1.2] for details).

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 4.6 and 4.8, 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 4.9. For the permutation that obtains C_2 from C_1, this follows from Proposition 4.10, and from the fact that the above bijection determines the permutation according to a strongly-explicit expander.

Propositions 4.6 and 4.8 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 LTCs with quasi-polylogarithmic query complexity

In this section we prove Theorem 1.2, which gives an infinite family of high-rate LTCs with quasi-polylogarithmic query complexity. The codes in this family also have the property of approaching the Singleton bound over constant size alphabet. This immediately proves Theorem 1.2 from the introduction.

Theorem 1.2. *For every $r\in(0,1)$ and every $\varepsilon>0$, there exists an explicit infinite family of linear codes $\{C_n\}_n$ satisfying:*

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

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

By concatenating the codes given by the above theorem with binary Gilbert-Varshamov codes [Gil52, Var57] of constant block length, we get the following stronger version of Theorem 1.1, which gives binary LTCs with quasi-polylogarithmic query complexity that attain the Zyablov bound [Zya71] (this immediately implies Theorem 1.1.

Theorem 5.1. *For every $r\in(0,1)$ and $\varepsilon>0$, there exists an explicit infinite family of binary linear codes $\{C_n\}_n$ satisfying:*

1. *C_n has block length at least n, rate at least r, and relative distance at least*

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

   where H^{-1} is the inverse of the binary entropy function in the domain $\left(0,\frac{1}{2}\right)$.

2. *C_n is locally testable with query complexity $(\log n)^{O(\log\log n)}$.*

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

We prove Theorem 1.2 in two stages: In the first stage, we construct LTCs with the desired query complexity that have *sub-constant* relative distance. In the second stage, we apply the Alon-Luby distance-amplification technique to the latter codes to obtain LTCs with the desired parameters. The first stage is the main part of the proof, and is formalized in the following result.

Lemma 5.2 (Main lemma)**.** *There exists an explicit infinite family of binary linear codes* $\{W_n\}_n$ *satisfying:*

1. W_n *has block length at least* n*, rate at least* $1-O(\frac{1}{\log n})$*, and relative distance at least* $\frac{1}{\log^{O(1)} n}$.
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* $(\log n)^{O(\log\log n)}$.

We prove the main lemma in Section 5.1. We now prove Theorem 1.2 based on this lemma.

Proof of Theorem 1.2 We would like to construct the desired LTCs by amplifying the relative distance of the LTCs of the main lemma. The straightforward solution would be to apply the Alon-Luby 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 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:

1. We start by amplifying the LTCs of the main lemma 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 $1-\mathrm{poly}(\varepsilon)$.
2. 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 $1-\mathrm{poly}(\varepsilon)$.
3. 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 5.2. 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 2.4) 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 $O(\varepsilon^3)$, 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 4.5), and thus we can apply Proposition 4.6.

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 $\exp(\text{poly}(1/\varepsilon))$, 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. ■

5.1 Proof of Lemma 5.2

As explained in the introduction, we construct the code W_n as follows: We start with a code of block length $\text{poly}\log(n)$, rate $1 - \frac{1}{\text{poly}\log(n)}$, and relative distance $\frac{1}{\text{poly}\log(n)}$. This code is trivially locally testable using $\text{poly}\log(n)$ 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 $\text{poly}\log(n)$ in each iteration, and the rate is roughy squared (actually, it is squared and then multiplied by $1 - \frac{1}{\text{poly}\log(n)}$). Hence, after about $\log \log n$ iterations, we end up with a code of block length n, rate $1 - \frac{1}{\text{poly}\log(n)}$, relative distance $\frac{1}{\text{poly}\log(n)}$, 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 $W \stackrel{\text{def}}{=} W_n$ 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 $\delta \stackrel{\text{def}}{=} \frac{1}{(\log n)^{36}}$. We choose B_0 to be the Zyablov code[5] of Fact 2.4 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 $\text{poly}\log(n)$. For every $i \geq 1$, we choose B_i to be the result of applying the $\square$-distance amplification (Lemma 4.4) to $(B_{i-1})^2$ to amplify the relative distance from δ^2 to δ.

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

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

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 4.4 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 4.4, 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 5.4. *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 4.4 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 3.4, 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 4.4, 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.

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.

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

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

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

[BGH+06] Eli Ben-Sasson, Oded Goldreich, Prahladh Harsha, Madhu Sudan, and Salil P. Vadhan. Robust pcps of proximity, shorter pcps, and applications to coding. *SIAM J. Comput.*, 36(4):889–974, 2006.

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

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

[BSVW03] Eli Ben-Sasson, Madhu Sudan, Salil P. Vadhan, and Avi Wigderson. Randomness-efficient low degree tests and short pcps via epsilon-biased sets. In *Proceedings of the 35th Annual ACM Symposium on Theory of Computing, June 9-11, 2003, San Diego, CA, USA*, pages 612–621, 2003.

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

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

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

[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. Preliminary version in STOC 2006.

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

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

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

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

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

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

[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.

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

[HS00] Prahladh Harsha and Madhu Sudan. Small pcps with low query complexity. *Computational Complexity*, 9(3-4):157–201, 2000.

[KMRS15] Swastik Kopparty, Or Meir, Noga Ron-Zewi, and Shubhangi Saraf. Locally correctable and testable codes approaching the singleton bound. *Electronic Colloquium on Computational Complexity (ECCC)*, 2015.

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

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

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

[RVW00] Omer Reingold, Salil P. Vadhan, and Avi Wigderson. Entropy waves, the zig-zag graph product, and new constant-degree expanders and extractors. In *41st Annual Symposium on Foundations of Computer Science, FOCS 2000, 12-14 November 2000, Redondo Beach, California, USA*, pages 3–13, 2000.

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

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

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

[Vid15a] Michael Viderman. A combination of testability and decodability by tensor products. *Random Struct. Algorithms*, 46(3):572–598, 2015.

[Vid15b] Michael Viderman. Explicit strong LTCs with inverse poly-log rate and constant soundness. *Electronic Colloquium on Computational Complexity (ECCC)*, 20, 2015.

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