Electronic Colloquium on Computational Complexity, Report No. 45 (2022)

Relaxed Locally Decodable and Correctable Codes: Beyond Tensoring

Gil Cohen[*] Tal Yankovitz[†]

April 4, 2022

Abstract

In their highly influential paper, Ben-Sasson, Goldreich, Harsha, Sudan, and Vadhan (STOC 2004) introduced the notion of a relaxed locally decodable code (RLDC). Similarly to a locally decodable code (Katz-Trevisan; STOC 2000), the former admits access to any desired message symbol with only a few queries to a possibly corrupted codeword. An RLDC, however, is allowed to abort when identifying corruption. The natural analog to locally correctable codes, dubbed relaxed locally correctable codes (RLCC), was introduced by Gur, Ramnarayan and Rothblum (ITCS 2018) who constructed asymptotically-good length-n RLCC and RLDC with $(\log n)^{O(\log\log n)}$ queries.

In this work we construct asymptotically-good RLDC and RLCC with an improved query complexity of $(\log n)^{O(\log\log\log n)}$. To achieve this, we devise a mechanism–an alternative to the tensor product–that squares the length of a given code. Compared to the tensor product that was used by Gur *et al.* and by many other constructions, our mechanism is significantly more efficient in terms of rate deterioration, allowing us to obtain our improved construction.

[*]Tel Aviv University. `gil@tauex.tau.ac.il`. Supported by ERC starting grant 949499 and by the Israel Science Foundation grant 1569/18.

[†]Tel Aviv University. `talyankovitz@mail.tau.ac.il`. Supported by ERC starting grant 949499.

ISSN 1433-8092

Contents

1 Introduction

Error correcting codes with "local structure" play a major role in modern coding theory, and their study is highly motivated by applications to theoretical computer science. Of particular interest are locally decodable codes (LDC), introduced by Katz and Trevisan [KT00], and locally correctable codes (LCC) that originated in works on program checking [BK95, Lip90]. These are error correcting codes that admit highly efficient procedures for recovering a single data symbol. LDC allow one to decode a specific symbol of the message while querying only a small number of symbols of the received, possibly corrupted, codeword. LCC on the other hand are equipped with a procedure for recovering any desired symbol of the codeword using few queries.

In their highly influential work, Ben-Sasson, Goldreich, Harsha, Sudan and Vadhan [BSGH+06] introduced a natural relaxation of LDC dubbed *relaxed locally decodable codes* (RLDC). Roughly speaking, the decoder of an RLDC is allowed to abort when identifying corruption, though it is required to always succeed when given access to a codeword. The natural analog of *relaxed locally correctable codes* (RLCC) was introduced by Gur, Ramnarayan and Rothblum [GRR20]. For linear codes, RLCC directly induce RLDC, and so in this case it can be easily seen that RLCC are stronger objects (for the case of non-linear codes, see [BGT16], Theorem A.6).

LDC, LCC and their relaxed counterparts have attracted significant attention (see [GI05, Woo07, Yek08, KY09, KV10, DGY11, Efr12, GKS13, KSY14, Mei14, HOW15, GG16, BGT16, GKO+18, LW19, GGK19, GRR20, CGS20, DGGW20, GL21, AS21, AG21, CY21b, CY21a, DGL21, BGGZ21] and references therein), and have found applications to PCPs [MR08, DH13, RZR20], property testing [CG18], privacy [GKST02], and probabilistic proof systems [GG21, GR17, GR18], to name a few.

For simplicity, in this introductory part we focus on binary codes. Formally, a (q, δ, ε) RLCC is an error correcting code $C \subseteq \{0,1\}^n$ that is equipped with a randomized "correction procedure" $\mathsf{Cor} : \{0,1\}^n \times [n] \to \{0,1\} \cup \{\perp\}$ having the following guarantees:

1. For every codeword $x \in C$, $\mathsf{Cor}(x, i) = x_i$ for every $i \in [n]$, with certainty.

2. For every $x \in \{0,1\}^n$ of distance at most δn from some codeword $y \in C$, and for every $i \in [n]$, $\mathsf{Cor}(x, i) \in \{y_i, \perp\}$ with probability at least $1 - \varepsilon$.

3. Cor makes at most q queries.

In this paper we consider asymptotically-good RLCC. The reader may consult [CGS20], and references therein, to learn more about the constant query regime. Gur, Ramnarayan and Rothblum [GRR20] constructed asymptotically-good RLCC with query complexity

$(\log n)^{O(\log\log n)}$. This offers a significant saving over the query complexity of the state-of-the-art LCC and LDC, $q = 2^{\tilde{O}(\sqrt{\log n})}$, obtained by Kopparty, Meir, Ron-Zewi, and Saraf [KMRS17]. As the [GRR20] construction is linear, it also yields an RLDC.

Interestingly, the idea underlying the Gur *et al.* construction is influenced by the construction of locally testable codes (LTC) of Kopparty *et al.* and not by the construction of LCC that appears in the same paper. Indeed, Kopparty *et al.* devised an LTC with query complexity $q = (\log n)^{O(\log\log n)}$ which stood as the state-of-the-art asymptotically-good LTC construction until the recent breakthrough by Dinur, Evra, Livne, Lubotzky, and Mozes [DEL$^+$21], and independently by Panteleev and Kalachev [PK21], that obtained asymptotically-good *constant* query LTC.

Answering a problem raised by Goldreich [Gol11], a beautiful result by Gur and Lachish [GL21] and by Dall'Agnol, Gur and Lachish [DGL21] rules out any hope of obtaining asymptotically-good RLCC with constant query complexity. Indeed, the lower bound obtained is $q = \Omega(\sqrt{\log n})$.

These results determine that the "playfield" for the query complexity of RLDC and RLCC lies in the exponent. That is, a main challenge in the area is to understand what is the least $e = e(n)$ for which a length-n asymptotically-good RLDC (or RLCC) with $(\log n)^{e(n)}$ queries exists [1]. Summarizing the above discussion, prior to this work it was known that

$$\frac{1}{2} \leqslant e(n) \leqslant c \cdot \log\log n,$$

for some constant $c \geqslant 1$. Naturally, for upper bounds on $e(n)$, an explicit construction is much preferred.

1.1 Our contribution

In this work we make progress on the study of relaxed locally decodable and correctable codes by constructing explicit asymptotically-good linear RLCC (and, by extension, RLDC) with significantly lower query complexity. Our result, stated below as Theorem 1.1, gives an exponential improvement in $e(n)$, achieving $e(n) = O(\log\log\log n)$.

Theorem 1.1 (Main result; informal)**.** *There are explicit asymptotically-good length-n linear RLCC (and thus RLDC) with query complexity*

$$q = (\log n)^{O(\log\log\log n)}.$$

The complete and formal statement of our result is given by Theorem 5.4.

[1]For the sake of brevity, we sometimes refer to the block-length of a code simply as its "length". Moreover, we alternate freely between the term distance and (local) error correction radius.

Perhaps no less important than the improvement itself is the underlying technique that we introduce. The construction of [GRR20] as well as many other constructions in the literature are based on the tensor product. This is a procedure that transforms a length-n code to a length-n^2 code with a manageable deterioration in the code parameters. As such the procedure serves as the basis for iterative constructions, starting with a small code that can be found, say, in a brute force manner, and ending up with a code with the desired length in $\log\log n$ iterations. In different settings, different ideas are required so to manage the deterioration of the code parameters throughout the iterations.

Our key idea lies in the construction of a new mechanism for enlarging the block-length from n to n^2. Our mechanism has a better deterioration of parameters compared to tensoring which in turn allows us to obtain our improved construction as given by Theorem 1.1. We believe that our technique may find other applications in some of the many other settings in which tensoring is used.

In Section 2 we give a detailed yet informal account of our "length squaring" mechanism and how it fits into our RLCC construction.

2 Proof technique

In this section we informally discuss our proof technique and, in particular, the underlying new mechanism which replaces the useful, yet expensive, tensor product in our RLCC construction. For simplicity we only consider binary codes though our construction works over any field. To begin with, in Section 2.1, we give an informal presentation of the tensor-based construction by Gur, Ramnarayan and Rothblum [GRR20].

2.1 The Gur-Ramnarayan-Rothblum construction

Following Kopparty *et al.*[KMRS17], the construction of a length-n RLCC by Gur, Ramnarayan and Rothblum [GRR20] is combinatorial in nature. Starting with an RLCC having a small block-length, the construction proceeds by enlarging the block-length in an iterative manner. At each iteration, a new code is obtained from the previous one, where the block-length increases at the expense of some deterioration of the remaining parameters, namely, the distance, rate, and query complexity (as well as the error parameter ε that we choose to suppress). At the end of each iteration, the distance of the new code is amplified back to its original value.

Ingredients. To give a more precise description of the construction we recall its two main ingredients:

1. Given a code $C \subseteq \{0,1\}^n$, we define the tensor product of C with itself, denoted $C \otimes C \subseteq \{0,1\}^{n^2}$, as follows. We think of the index set of $C \otimes C$ as $[n] \times [n]$ and define $C \otimes C$ to be the code that consists of all $n \times n$ binary matrices having the following property: every row and every column of the matrix is a codeword of C. It is well-known that the rate of $C \otimes C$ is at least ρ^2, where ρ is the rate of C.

2. The second ingredient is a *distance amplification procedure*. This is a procedure that amplifies the distance of a given code or, more precisely, the local error correction radius in our case, without significant deterioration in the remaining parameters. In particular, for local codes such as LCC, LDC and their relaxed counterparts, the crucial point is to minimize the deterioration in the query complexity. Starting with the works of Alon, Edmonds, and Luby [AEL95, AL96], several works obtained new and improved distance amplification procedures [KMRS17, CY21b, CY21a].

   For this informal introductory part, it suffices to know that a distance-δ' linear binary RLCC can be transformed to a distance $\delta > \delta'$ linear binary RLCC with an additive rate deterioration of $\mathrm{poly}(\delta)$ and a multiplicative cost of $\mathrm{poly}(\frac{1}{\delta'})$ to the query complexity.

The Gur-Ramnarayan-Rothblum RLCC. With the above-mentioned ingredients, we can present the Gur-Ramnarayan-Rothblum RLCC construction.

1. Start with a small RLCC $C_0 \subseteq \{0,1\}^{n_0}$.
2. For $j = 1, \dots, t = \log\log n$
   (a) Set $C'_j = C_{j-1} \otimes C_{j-1} \subseteq \{0,1\}^{n_j}$, where $n_j = n_{j-1}^2$.
   (b) Set $C_j = \mathsf{DistAmp}(C'_j) \subseteq \{0,1\}^{n_j}$, where $\mathsf{DistAmp}$ is set so that the distance of C_j matches the distance of C_{j-1}.
3. Return $\mathsf{DistAmp}(C_t)$, where $\mathsf{DistAmp}$ amplifies the distance to a constant.

Rate deterioration. In the above construction, at each iteration the block-length squares in Step (2a) via the tensor product. Thus, in $\log\log n$ iterations one would get a length-n (or a slightly larger) code. This progress has its cost, and we first consider the effect on the rate. As mentioned, if C_{j-1} has rate ρ_{j-1} then the rate of C'_j is ρ_{j-1}^2. There is an additional cost to the rate incurred by the distance amplification procedure in Step (2b) though even if we ignore it, we see that the rate of C_j satisfies $\rho_j \leqslant \rho_{j-1}^2$.

This results with a final rate $\rho_t \leqslant \rho_0^{2^t}$. Thus, the only way of ending up with a constant rate code C_t is to set the base code C_0 to have rate

$$\rho_0 \approx 1 - 2^{-t} = 1 - \frac{1}{\log n}.$$

By, say, the Singleton bound, this forces the distance δ_0 of C_0 to be at most $\frac{1}{\log n}$. Further, we see that n_0, the base code's length, must be taken at least logarithmic in n.

Query deterioration and the error correction radius. On top of its rate deterioration, the tensor product also squares the error correction radius. Letting δ_{j-1} be the error correction radius of C_{j-1}, it is known that C'_j has error correction radius δ_{j-1}^2. In Step (2b) the latter is amplified back to δ_{j-1} at a small cost in rate, which we ignore. Thus, the error correction radius remains δ_0 throughout.

Both the tensor product and the distance amplification procedure have a $\text{poly}(\frac{1}{\delta_{j-1}}) = \text{poly}(\frac{1}{\delta_0})$ multiplicative cost in query complexity. Denoting the query complexity of C_j by q_j, we get the recurrence relation

$$q_j = q_{j-1} \cdot \text{poly}\frac{1}{\delta_0}, \tag{2.1}$$

resulting with $q_t = (\frac{1}{\delta_0})^{O(\log\log n)}$. But now recall that the rate deterioration forces $\delta_0 \leqslant \frac{1}{\log n}$ and so the query complexity incurs a multiplicative $\text{poly}(\log n)$ cost in each iteration, resulting with a final query complexity of $q = (\log n)^{\Theta(\log\log n)}$.

Note that throughout the iterations we maintain a fixed sub-constant distance $\delta_0 \approx \frac{1}{\log n}$. We cannot afford to work with constant distance at the iterative stage because we must keep the rate high (we do not have a sufficiently good *rate* amplification procedure; see [CY21b]). However, once the iterative process terminates and we have obtained the desired block-length, at Step (3) we can afford to amplify the distance, one last time, to a constant.

The tensor-based corrector. To see why the tensor product causes a query deterioration as given by Equation (2.1), we take a look at how a local corrector for $C'_j = C_j \otimes C_j$ is defined given a local corrector for C_j. For correcting entry $(a, b) \in [n_j]^2$ the corrector proceeds by first invoking the corrector for C_j to the a^{th} row, querying for index b to obtain $\sigma \in \{0, 1, \perp\}$. If $\sigma = \perp$, the corrector safely returns $\perp$ as well. But, of course, the a^{th} row accounts for a negligible fraction of the codeword and so an adversary can easily alter that row to a different codeword, resulting with an incorrect $\sigma \in \{0, 1\}$.

Therefore, the corrector for C_j must have a mechanism for measuring how confident it is in σ being the correct bit. To this end, we only need to consider the problematic case in which the a^{th} row has more than a $\delta \triangleq \delta_0 = \delta_j$ fraction of errors. The corrector will make use of the fact that each column is also an RLCC so to try and identify that this is the case. To do so, the corrector samples $\Theta(\frac{1}{\delta})$ indices in the a^{th} row uniformly and independently at random. As we are in the problematic case, we will hit a corrupted bit with good probability. For each such sample–a point (a, c)–the corrector will utilize the RLCC that is embedded in the c^{th} column to retrieve entry a. If we denote the result returned by $\tau_c \in \{0, 1, \perp\}$ then the corrector will safely return $\perp$ if $\tau_c = \perp$. Otherwise, it will compare τ_c against the bit in entry (a, c), and will return $\perp$ in the case of a mismatch.

As these samples are chosen uniformly and independently at random, a simple averaging argument can be used to show that a column c with corruption at (a, c), and not too many corruptions otherwise will be considered. As a result, a mismatch between the resulted query and the bit read directly will lead the corrector to return $\perp$ in this problematic case, with high probability.

2.2 The double cost of tensoring

By inspecting the tensor-based construction discussed in Section 2.1, we see that the quadratic rate deterioration of the tensor product requires us to start with a rate $1 - \frac{1}{\log n}$ base code C_0 which, in turn, forces the distance to be bounded by $\delta \leqslant \frac{1}{\log n}$. As the query complexity suffers a multiplicative $\mathrm{poly}(\frac{1}{\delta}) = \mathrm{poly}\log n$ in each iteration, we get the above-mentioned query complexity.

Thus, a natural idea is to try and come up with a more rate-efficient procedure–an alternative to tensoring–that increases the length of the code, hopefully squares it. Indeed the rate deterioration is "responsible" for the deterioration of the remaining parameters.

One first needs to understand why does tensoring squares the rate. Consider a code $C \subseteq \{0, 1\}^n$ with rate ρ. As the rates we consider are close to 1, we write $\rho = 1 - \alpha$ where the "rate deficiency" α should be thought of as small, say, $\alpha = o(1)$. While one can show that the rate of $C \otimes C$ is $\rho^2 = 1 - 2\alpha + \alpha^2$, for our purpose, it will be useful to describe a simple argument that guarantees a slightly lower rate of $1 - 2\alpha$. Note that the difference between the two bounds is negligible in our regime of parameters. To see that the rate is such, recall that C is determined by αn linear constraints. Therefore, in $C \otimes C$ one introduces αn linear constraints for each row and for each column. This adds up to $2n \cdot \alpha n$ linear constraints, resulting with rate $1 - 2\alpha$.

Observe that the reason we need the rows and columns to be RLCC is very different. Indeed, a row being RLCC allows us to retrieve entry (a, b) by using the a^{th} row's RLCC-ness to retrieve the bit at index b, obtaining a result $\sigma \in \{0, 1, \perp\}$. Had $\sigma \neq \perp$, the columns's RLCC-ness are used to vet for the correctness of σ.

2.3 Our RLCC

Our key idea lies in introducing a mechanism that allows us to drop the requirement of the columns being RLCC. Instead we use the fact that the rows are RLCC *both* for the first step of retrieving the desired bit as well as for the second step of vetting for the correctness of the retrieved bit. Of course, our mechanism also has a cost in rate but it is significantly more economical. Instead of doubling the rate deficiency, moving from $1 - \alpha$ to $1 - 2\alpha$, we only incur an additive loss of β resulting in rate $1 - \alpha - \beta$. Thinking ahead of the iterative process, this smaller loss yields a linear (rather than exponential) deterioration throughout the iterations, which enables us to take the initial α and β reciprocal to the number of iterations. This then enables the query complexity to deteriorate more slowly, but we are getting ahead of ourselves.

Row-evasive partitions. An ingredient we need for our construction is an object we dub a *row-evasive partition*. This is a set of n partitions of $[n]$, $\mathcal{P} = \{P_1, \dots, P_n\}$, where each part of each partition is of size b for some parameter b to be set later on. We think of P_j as a partition of the j^{th} column of $[n] \times [n]$ and require the collection to be row-evasive in the following sense: For all distinct $i, i' \in [n]$,

$$|\{j \in [n] \mid i' \in (P_j)_i\}| \leqslant 2b, \tag{2.2}$$

where $(P_j)_i$ is the set of elements in the part containing i, excluding i itself, in partition P_j. From here on we denote $\beta = \frac{1}{b}$. Informally, the partition is row-evasive if every two distinct rows have few parts that intersect both rows.

In this section we leave the question of the existence and, by extension, the construction of row-evasive partitions and assume such is available to us. Our explicit construction of row-evasive partitions can be found in Section 4.1.

Squaring without tensoring. We are now ready to present our mechanism for squaring the block-length. We in fact present a restricted version of it which suffices for the proof of Theorem 1.1. However, a more general and arguably more natural (though somewhat more technically involved) mechanism is covered in the formal part of this paper (see Section 4.2).

Given a code $C \subseteq \{0,1\}^n$ we define the code $C_{\mathcal{P}} \subseteq \{0,1\}^{n^2}$, $\mathcal{P}$ being a row-evasive partition of size $|\mathcal{P}| = n$ as follows. Assume C has rate $1 - \alpha$. Then, C is defined by αn linear constraints. Thinking of the indices of $C_{\mathcal{P}}$ as $[n] \times [n]$, the code $C_{\mathcal{P}}$ is defined by enforcing these constraints at each row. Moreover, for each part of each partition of $\mathcal{P}$ we introduce the linear constraint enforcing the parity of the bits in that part to vanish [2]. Thus, we have αn constraints per row and $\frac{n}{b} = \beta n$ constraints per column, giving rise to a total number of

$$\alpha n \cdot n + \beta n \cdot n = (\alpha + \beta) n^2$$

constraints. Hence, the rate of $C_{\mathcal{P}}$ is at least $1 - \alpha - \beta$, and so the rate-loss incurred by moving from C to $C_{\mathcal{P}}$ is only an additive β. As discussed above, the reader should compare this to the factor-2 multiplicative loss of the tensor product $C \otimes C$ in the high rate regime.

The corrector for $C_{\mathcal{P}}$. We turn to describe the corrector for $C_{\mathcal{P}}$. Say we wish to query entry $(a, b) \in [n] \times [n]$. Much like the corrector for the tensor product that was described in Section 2.1, our corrector also makes use of the row to query the desired entry and then, upon receiving a non $\perp$ symbol, proceeds by vetting the result before returning it.

As in the analysis of the tensor product, we only need to consider the case in which the a^{th} row has more corruptions than the code that is embedded in that row can handle by itself, namely, more than δn corruptions. We similarly proceed by sampling some number of "vetting indices" in the row, querying each directly and compare the result against a more global process, hoping to identify a mismatch. However, whereas in the tensor-based construction this global process invokes the RLCC-ness of the columns, we do not longer have that option available to us, and this is where our corrector deviates from the tensor-based corrector.

For each of the vetting indices in row a we proceed as follows. Consider a vetting index $(a, j) \in [n] \times [n]$. We first directly read entry (a, j) as well as the other $b - 1$ indices in $(P_j)_a$ - the part of partition P_j that contains a. Recall that the parity of these bits must vanish in a codeword, and so if this is not the case, we can safely return $\perp$. For each $i \in (P_j)_a$, using the RLCC-ness of the i^{th} row, we query index j and compare it against the bit we read directly. In case of a mismatch we return $\perp$; otherwise, if no mismatch have occurred, we return the vetted bit.

Error analysis of the corrector for $C_{\mathcal{P}}$. Say that C has error correction radius δ and that we are aiming for error correction radius δ' to be determined later on. For comparison, recall that for the tensor product it is known that $\delta' = \delta^2$. Let B be the set of rows in which more than a δ-fraction of corruptions have occurred. By a simple averaging argument, $|B| \leqslant \frac{\delta'}{\delta} n$.

[2]The more general row-evasive squaring that we consider in the formal sections of this paper, uses not just a single parity check on the bits of each part but rather parity checks as dictated by a small RLCC.

As mentioned, we focus on the case where row a has more than a δ-fraction of corrupted bits. Let $J \subseteq [n]$ be the set of corrupted indices in that row. Further, let $J' \subseteq [n]$ be the set of indices in row a for which the corresponding parts of $\mathcal{P}$ intersect B. That is,

$$J' = \{j \in [n] \mid (P_j)_a \cap B \neq \varnothing\} . \tag{2.3}$$

Observe that any $j \in J \backslash J'$ will be a successful vetting index. This is because all points in $(P_j)_a$ lie in rows that have fewer than a δ-fraction of errors, and so all the $b-1$ applications of the corresponding RLCC will succeed with high probability. Therefore, as $j \in J$, this will result in a mismatch with high probability.

By the above observation, the task at hand is to bound the size of $J \backslash J'$ from below, but recall that $|J| \geqslant \delta n$, and so it suffices to bound $|J'|$ from above by, say, $\frac{\delta n}{2}$. This is where the row-evasiveness comes into play. Indeed, by Equation (2.3),

$$|J'| \leqslant \sum_{j \in J'} |(P_j)_a \cap B|.$$

Now, if we switch to count via the rows of $B \backslash \{a\}$ rather than the columns of J', we get

$$\sum_{j \in J'} |(P_j)_a \cap B| = \sum_{i \in B \backslash \{a\}} |\{j \in J' \mid i \in (P_j)_a\}|,$$

and so

$$|J'| \leqslant \sum_{i \in B \backslash \{a\}} |\{j \in [n] \mid i \in (P_j)_a\}|.$$

By the defining property of $\mathcal{P}$ as given by Equation (2.2), for each $i \neq a$,

$$|\{j \in [n] \mid i \in (P_j)_a\}| \leqslant 2b,$$

and so

$$|J'| < 2b \cdot |B| \leqslant 2b \cdot \frac{\delta'}{\delta} n.$$

Setting $\delta' = \frac{\delta^2}{4b}$ we can bound $|J'| < \frac{\delta}{2} n$ and so $|J \backslash J'| > \frac{\delta}{2} n$.

To recap, every index in $J \backslash J'$ is a good vetting index, and $|J \backslash J'| \geqslant \frac{\delta n}{2}$. Hence, by sampling, say, $\frac{2}{\delta}$ vetting indices we will find a good vetting index with good probability. For comparison, recall that the tensor product has distance deterioration of $\delta' = \delta^2$ whereas we suffer a slightly larger deterioration as $\delta' = \frac{\delta^2}{4b}$. This turns out to be a nonissue as the distance amplification step that will follow works just as well, with the same asymptotic cost in query complexity for our setting of the parameter b.

The query complexity of the corrector for $C_{\mathcal{P}}$. Recall that we first use the RLCC-ness of the desired row, which amounts to q queries. Additionally, the vetting process then begins by sampling $\frac{2}{\delta}$ vetting indices in that row; for each, querying the corresponding part, and for each entry in each part invoking a row corrector. This results in an overall number of $O(\frac{bq}{\delta})$ queries.

In comparison, the query complexity resulted by the tensor product is $O(\frac{q}{\delta})$. Thus, like in the case of the distance deterioration, our mechanism is slightly more costly in terms of query complexity. However, this turns out to be negligible compared to the cost in query complexity that results from the significantly higher rate deterioration of the tensor product compared to our mechanism.

The RLCC construction. We now plug this machinery that we developed as a replacement for the tensor product into the framework of [GRR20] to get our RLCC construction.

1. Start with a small RLCC $C_0 \subseteq \{0,1\}^{n_0}$ with distance δ.
2. For $j = 1, \ldots, t = \log\log n$
   (a) Let $\mathcal{P}_{j-1}$ be a row-evasive partition, $|\mathcal{P}_{j-1}| = n_{j-1}$, with parts of size b.
   (b) Set $C'_j = (C_{j-1})_{\mathcal{P}_{j-1}} \subseteq \{0,1\}^{n_j}$, and note that $n_j = n_{j-1}^2$.
   (c) Set $C_j = \mathsf{DistAmp}(C'_j) \subseteq \{0,1\}^{n_j}$ so that the distance is amplified back to match the distance of C_{j-1}, namely, to distance δ.
3. Return $\mathsf{DistAmp}(C_t)$, where $\mathsf{DistAmp}$ is set so that the distance is amplified to a constant.

Analyzing the RLCC construction. Recall that the rate deterioration is "responsible" for the deterioration of all other parameters. So we start by considering the rate. If we denote the rate of C_j by $1-\alpha_j$, and the rate of C'_j by $1-\alpha'_j$ then, by the above discussion,

$$\alpha'_j \leqslant \alpha_{j-1} + \beta.$$

Taking into account the rate deterioration of the distance amplification procedure, we have that

$$\alpha_j \leqslant \alpha'_j + \mathrm{poly}(\delta).$$

Thus, $\alpha_j \leqslant \beta + \mathrm{poly}(\delta) + \alpha_{j-1}$, and so

$$\alpha_t \leqslant t \cdot (\beta + \mathrm{poly}(\delta)) + \alpha_0. \tag{2.4}$$

With this in mind, it is time to set the parameters. To ensure that C_t has constant rate, since $t = \log\log n$, we set $\beta = O(\frac{1}{\log\log n})$. Next, we take the base code C_0 to have distance $\delta = \mathrm{poly}\left(\frac{1}{\log\log n}\right)$ so that the contribution of the $\mathrm{poly}(\delta)$ term in Equation (2.4) is also $O(\frac{1}{\log\log n})$. Recall now that C_0 has rate $1-\alpha_0$. As we set C_0 to have distance δ we are, of course, not free to set α_0 as we wish. However, we can take $\alpha_0 = \mathrm{poly}(\delta) = \mathrm{poly}\left(\frac{1}{\log\log n}\right)$. This choice of parameters ensures that C_t has indeed constant rate.

We now consider the effect of this choice of parameters to the query complexity. Denote the query complexity of C_j by q_j. Then, the query complexity of C'_j is of the order of

$$\frac{b \cdot q_{j-1}}{\delta} = q_{j-1} \cdot \mathrm{poly}(\log\log n).$$

As the distance amplification procedure increase the query complexity by a multiplicative $\mathrm{poly}(\frac{1}{\delta'})$ factor, where δ' is the distance of C'_j, and since $\delta' = \mathrm{poly}(\delta) = \mathrm{poly}\left(\frac{1}{\log\log n}\right)$, we have that

$$q_j = q_{j-1} \cdot \mathrm{poly}(\log\log n),$$

and so C_t has query complexity

$$q_t = (\log\log n)^{O(\log\log n)} = (\log n)^{O(\log\log\log n)},$$

which is also the query complexity of $\mathsf{DistAmp}(C_t)$ - the code obtained after the last application of the distance amplification procedure.

Comparison with axis-evasive partitions. Cohen and Yankovitz devised a rate amplification procedure for LCC [CY21b]. More precisely, their procedure efficiently transforms a q-query LCC with rate ρ to a constant rate LCC with query complexity $q^{\mathrm{poly}(1/\rho)}$.

A key idea in their construction is to use the rate deterioration of the tensor product to their *favor*. Indeed, the authors took the tensor product of the *dual* codes, thus decreasing the dimension of the dual code as a way of increasing the dimension of the code. This however has a significant cost in distance. To overcome this, the authors considered a pseudorandom partition of the coordinate set $[n] \times [n]$ and, for each part in the partition, added a dual constraint requiring the parity of the bits in each part to vanish (slightly harming the rate). This allowed them to read a bit by querying the other pseudo-randomly located indices in its part. The pseudorandom notion they required, dubbed *axis-evasive partition*, is tailored to their need.

Our construction is influenced by this idea of adding dual constraints that are induced by pseudorandom partitions. Of course, the setting is not the same and our requirement of the partition is different. Indeed, our use of the partitions is not for distance amplification but rather as a way of reusing the row RLCC-ness to vet for a potential output bit.

3 Preliminaries

3.1 Notations and conventions

Unless stated otherwise, all logarithms are taken to the base 2. The set of natural numbers is $\mathbb{N} = \{0, 1, 2, \ldots\}$ and we also use $\mathbb{N}_{\text{even}} = \{0, 2, 4, \ldots\}$ and $\mathbb{N}_{\text{odd}} = \{1, 3, 5, \ldots\}$. For $n \in \mathbb{N}$, $n \geqslant 1$, we use $[n]$ to denote the set $\{1, \ldots, n\}$. For ease of readability, we sometimes avoid the use of floor and ceiling. This does not affect the stated results. For $q \in \mathbb{N}$, $q \geqslant 2$, we use H_q to denote the q-ary entropy function, and $\mathrm{H} = \mathrm{H}_2$ to denote the binary entropy function. We use $\mathbb{F}$ to denote a field, and any referenced field is assumed to be finite.

For a finite set N, we refer to a function $v \in \mathbb{F}^N$ as a *vector* and we say that it is *indexed by* N. For a vector $v \in \mathbb{F}^N$ and $i \in N$ we use v_i as a shorthand for $v(i)$. For a vector $v \in \mathbb{F}^N$ and a set $N' \subseteq N$ we denote by $v_{N'}$ the vector $v' \in \mathbb{F}^{N'}$ such that $v'_i = v_i$ for every $i \in N'$. For two vectors $u, v \in \mathbb{F}^N$, their (absolute) *hamming distance* is $|\{i \in N \mid u_i \neq v_i\}|$, which we denote by $\mathsf{dist}(u, v)$, and the *hamming weight* of u is $\mathsf{dist}(u, \bar{0})$. For two vectors $u, v \in \mathbb{F}^N$, their *inner product* is $\sum_{i \in N} u_i v_i$ (over $\mathbb{F}$) and we denote it by $\langle u, v \rangle$.

3.2 Entropy related inequalities

We will make use of the following easily verified facts.

Fact 3.1. *For every $q \in \mathbb{N}$, $q \geqslant 2$, and $\Delta \in \left(0, \frac{1}{2}\right]$, $\mathrm{H}_q(\Delta) \leqslant 2\Delta \log \frac{1}{\Delta}$.*

Fact 3.2. *For every $q \in \mathbb{N}$, $q \geqslant 2$, and $0 \leqslant \Delta \leqslant 1 - \frac{1}{q}$, it holds that $\mathrm{H}_q(\Delta) \geqslant \Delta$.*

Fact 3.3. *For every $q \in \mathbb{N}$, $q \geqslant 2$, $c \geqslant 1$ and $0 \leqslant \Delta \leqslant \frac{1}{c}$, it holds that $c \cdot \mathrm{H}_q(\Delta) \geqslant \mathrm{H}_q(c \cdot \Delta)$.*

3.3 Error correcting codes

We start by recalling the definition of an error correcting code. In this work we only consider linear codes. The definition below is standard, however, for our purposes we find it convenient to work with an arbitrary index set rather than the usual set $[n]$, and so the reader may benefit from glancing over the definition.

Definition 3.4. *For a finite set N of size $|N| = n$ and a field $\mathbb{F}$, a* code *is a linear subspace $C \subseteq \mathbb{F}^N$. We say that the code C is* indexed *by N and that it is* over *$\mathbb{F}$. The* length *of the code is n. The* dimension *of the code, usually denoted by k, is the dimension of C over $\mathbb{F}$, $\dim_{\mathbb{F}} C$. The (non-local) distance of the code, denoted by d, is $\min_{c, c' \in C, c \neq c'} \mathsf{dist}(c, c')$. The rate of the code, typically denoted by ρ, is $\frac{k}{n}$. The (non-local) relative distance of the code, denoted by Δ, is defined to be $\frac{d}{n}$. The elements of C are called* codewords.

The following is a straightforward claim about the dimension of linear codes.

Claim 3.5. *Let $C \subseteq \mathbb{F}^N$ and $C' \subseteq \mathbb{F}^N$ be two codes, with rates ρ and ρ', respectively. Then, $C \cap C'$ has rate at least $\rho + \rho' - 1$.*

3.4 Relaxed locally correctable codes

We turn to recall the definition of relaxed locally correctable codes as put forth by Gur, Ramnarayan and Rothblum [GRR20].

Definition 3.6. *A code $C \subseteq \mathbb{F}^N$ of length n is called a (q, δ, ε)-*RLCC *(relaxed locally correctable code, abbreviated) if there exists a randomized procedure*

$$\mathsf{Cor} : N \to \mathbb{F} \cup \{\perp\}$$

that is given an oracle access to $z \in \mathbb{F}^N$, and has the following guarantee. For every $i \in N$, $y \in C$ and $z \in \mathbb{F}^N$, satisfying $0 < \mathsf{dist}(z, y) \leqslant \delta n$, $\mathsf{Cor}^z(i) \in \{y_i, \perp\}$ with probability at least $1 - \varepsilon$; if $z = y$ (namely, $\mathsf{dist}(z, y) = 0$), we require that $\mathsf{Cor}^z(i) = y_i$, with probability 1. Furthermore, $\mathsf{Cor}^z(i)$ always makes at most q queries to z. We call Cor a local corrector *(or* corrector*). The parameter δ is called the* correction radius*, and the parameter q is called the* query complexity*.*

The error parameter of an RLCC can be easily amplified, with low cost to the query complexity, as stated in the following claim.

Claim 3.7. *Let $C \subseteq \mathbb{F}^N$ be a (q, δ, ε)-RLCC. Then, for any $h \in \mathbb{N}$, C is also an $(hq, \delta, \varepsilon^h)$-RLCC.*

Proof. Let Cor be a local corrector for C. For any $z \in \mathbb{F}^N$ and $c \in C$ such that $\mathsf{dist}(z, c) \leqslant \delta|N|$, and for any $i \in N$, $\mathsf{Cor}^z(i)$ outputs a symbol not in $\{c_i, \perp\}$ with probability less than ε. A correction procedure with a reduced error probability $\varepsilon' = \varepsilon^h$ is achieved simply by invoking $\mathsf{Cor}(i)$ for h times. If there is a disagreement between the outputs of the different runs of $\mathsf{Cor}(i)$, or if one the runs resulted in $\perp$, then it is certain that z is not a codeword and so $\perp$ can be outputted in such a case. If all the runs are in agreement (on a symbol other than $\perp$), then note that they can all agree on a wrong symbol only if all the runs failed, and the probability for that event is bounded by ε^h, as the runs are independent. It follows that the error probability of this corrector is indeed ε^h, and clearly the query complexity is hq, as required. $\square$

4 Row-evasive squaring

In this section we define the tensor-like mechanism for squaring the length of an RLCC while only moderately deteriorating its remaining parameters. In Section 4.1 we introduce the notion of row-evasive partitions which is a key component in this mechanism, and give an explicit construction of such partitions. Then, in Section 4.2 we describe the row-evasive squaring mechanism.

4.1 Row-evasive partitions

In this section we introduce the notion of row-evasive partitions, which are needed for the operation we put forth. Let P be a partition of a set N and let $i \in N$. We denote by $[P]_i$ the set of elements of the part to which i belongs, and let $(P)_i = [P]_i \backslash \{i\}$.

Definition 4.1. *Let N be a set. Let $\mathcal{P} = \{P_i \mid i \in N\}$ be a set of partitions of N into parts of size b. We say that $\mathcal{P}$ is* row-evasive *if for every distinct $i, i' \in N$,*

$$|\{j \in N \mid i' \in (P_j)_i\}| \leqslant 2b. \tag{4.1}$$

Claim 4.2. *For every N such that $|N| = n = 2^k$, where $k \in \mathbb{N}$, and every $\ell \leqslant k$, there exists a set of partitions $\mathcal{P} = \{P_i \mid i \in N\}$ of N into parts of size $b = 2^\ell$ that is row-evasive.*

Our proof of Claim 4.2 is based on simple modular arithmetic. While we could have based a somewhat cleaner construction on finite fields (also obtaining a slightly better bound of b rather than $2b$ in Equation (4.1)), we find it more convenient to work with the rings that we (implicitly) consider. Indeed, in the finite-field-based that we omit, the construction requires that N is of the same size as the multiplicative group of the field, which limits the range of partition size we can obtain. In particular, due to the iterative nature of our RLCC construction it is convenient to have a row-evasive partition for every size that is a power of two. Moreover, we note that any universal hash family [CW79], $H = \{h : [n] \to [m]\}$, can be used to construct a row-evasive partition, given that $|H|, n, m$ are appropriate and that it satisfies the following extra property: for every $h \in H$ and $r, r' \in [m]$, $|h^{-1}(r)| = |h^{-1}(r')|$.

Proof of Claim 4.2. We may assume without loss of generality that $N = \{0, \ldots, n-1\}$. Define for every $i, j \in N$,

$$f(i,j) = (2i+1)(2j+1) \mod 2^{k+1}.$$

Note that for every $i, j \in N$, it holds that $f(i,j) \in \{0, \ldots, 2^{k+1}-1\} \cap \mathbb{N}_{\text{odd}}$. Further define for every $m \in \{0, \ldots, 2^{k+1}-1\} \cap \mathbb{N}_{\text{odd}}$,

$$g(m) = m \mod 2^{k-\ell+1}.$$

Note that for every $m \in \{0, \ldots, 2^{k+1} - 1\} \cap \mathbb{N}_{\text{odd}}$, $g(m) \in \{0, \ldots, 2^{k-\ell+1} - 1\} \cap \mathbb{N}_{\text{odd}}$. Moreover, for every $r \in \{0, \ldots, 2^{k-\ell+1} - 1\} \cap \mathbb{N}_{\text{odd}}$

$$g^{-1}(r) = \left\{t \in \{0, \ldots, 2^{k+1} - 1\} \cap \mathbb{N}_{\text{odd}} \mid g(t) = r\right\},$$

and note that $|g^{-1}(r)| = \frac{2^{k+1}}{2^{k-\ell+1}} = b$, for every such r.

The set of partitions $\mathcal{P} = \{P_j \mid j \in N\}$ is constructed as follows. For every $j \in N$, the parts of partition P_j are given by

$$\{i \in N \mid g(f(i,j)) = r\}_{r \in \{0, \ldots, 2^{k-\ell+1}-1\} \cap \mathbb{N}_{\text{odd}}}.$$

To show that $\mathcal{P}$ is as required, we first need to show that every part of every P_j is of size b. This is indeed the case, as for every $j \in N$, and every $i, i' \in N$, $i \neq i'$, we have that $f(i,j) \neq f(i',j)$. This can be seen as

$$f(i,j) - f(i',j) \mod 2^{k+1} = (2i - 2i')(2j+1) \mod 2^{k+1} \neq 0.$$

The inequality follows since $(2j+1)$ is odd, then $2^{k+1} \mid (2i-2i')(2j+1) \mod 2^{k+1}$ implies $2^{k+1} \mid (2i - 2i')$. But as $i, i' \in \{0, \ldots, 2^k - 1\}$ and $i \neq i'$, we have that $0 < |2i - 2i'| \leqslant 2^{k+1} - 2$ so it cannot be that $2^{k+1} \mid (2i - 2i')$. It follows that for every $j \in N$, $f(\cdot, j)$ is a bijective mapping from $\{0, \ldots, 2^k - 1\}$ onto $\{0, \ldots, 2^{k+1} - 1\} \cap \mathbb{N}_{\text{odd}}$. Therefore, as for every $j \in N$, the elements of each part of P_j are $\{i \in N \mid g(f(i,j)) = r\}$ for some $r \in \{0, \ldots, 2^{k-\ell+1} - 1\} \cap \mathbb{N}_{\text{odd}}$, and $|g^{-1}(r)| = b$, we conclude that each part's size is indeed b.

Secondly, to prove that $\mathcal{P}$ satisfies the row-evasive requirement, we need to show that for every distinct $i, i' \in N$,

$$|\{j \in N \mid g(f(i,j)) = g(f(i',j))\}| \leqslant 2b.$$

Towards that, note that we have that if $g(f(i,j)) = g(f(i',j))$ then $f(i,j) - f(i',j) \mod 2^{k-\ell+1} = 0$, which implies in turn that

$$\left(2(i - i')(2j+1) \mod 2^{k+1}\right) \mod 2^{k-\ell+1} = 0.$$

Define for every $j \in N$,

$$h_{i,i'}(j) = 2(i - i')(2j+1) \mod 2^{k+1},$$

and note that for every $t \in \{0, \ldots, 2^{k+1} - 1\} \cap \mathbb{N}_{\text{even}}$ there are exactly two distinct $j, j' \in N$ such that $h_{i,i'}(j) = h_{i,i'}(j') = t$ (as there are exactly two solutions $x \in \{0, \ldots, 2^{k+1} - 1\}$ to the equation $2x \mod 2^{k+1} = 0$, namely, 0 and 2^k). Furthermore, there are exactly

$\frac{2^k}{2^{k-\ell}} = b$ values $t \in \{0, \ldots, 2^{k+1}-1\} \cap \mathbb{N}_{\text{even}}$ such that $g(t) = 0$, and let T denote the set of these values. It follows that

$$\begin{aligned}
|\{j \in N \mid g(f(i,j)) = g(f(i',j))\}| &\leqslant |\{j \in N \mid g(h_{i,i'}(j)) = 0\}| \\
&= \{j \in N \mid h_{i,i'}(j) \in T\} \\
&= 2|T| \\
&= 2b.
\end{aligned}$$

This shows that the requirement is met, and the claim follows. □

4.2 The row-evasive squaring

In this part we define the basic operation of the construction, we dub "row-evasive squaring".

Definition 4.3. *Let N be an arbitrary finite set and $b \geqslant 1$ an integer dividing $|N|$. Let $C_b \subseteq \mathbb{F}^{[b]}$ be a code and let P be a partition of N into $\frac{|N|}{b}$ parts of size b. For every $B \in P$, assume an arbitrary fixed bijection $f_B : B \to [b]$ and let $C_B \subseteq \mathbb{F}^B$ be the code $\{c \circ f \mid c \in C_b\}$. Define*

$$C^N_{P,C_b} = \left\{c \in \mathbb{F}^N \mid \forall B \in P \quad c_B \in C_B\right\}.$$

It is immediate that given that a code $C_b \subseteq \mathbb{F}^{[b]}$ is a (q, δ, ε)-RLCC then so is any C_B as defined in Definition 4.3.

Definition 4.4 (The squaring operation)**.** *Let N be an arbitrary finite set and $b \geqslant 1$ an integer dividing $|N|$. Let $C \subseteq \mathbb{F}^N$ be a code, let $C_b \subseteq \mathbb{F}^{[b]}$ be a code and let $\mathcal{P} = \{P_i \mid i \in N\}$ be a set of partitions of N into parts of size b. We define the $(\mathcal{P}, C_b)$-*squaring *of C, denoted by $(C)^2_{\mathcal{P},C_b}$, to be the code*

$$(C)^2_{\mathcal{P},C_b} = \left\{c \in \mathbb{F}^{N\times N} \mid \forall i \in N \quad c_{\{i\}\times N} \in C \quad \textit{and} \quad \forall j \in N \quad c_{N\times\{j\}} \in C^{N\times\{j\}}_{P_j,C_b}\right\}.$$

A bound on the rate of the row-evasive squaring of a code is given in the following claim.

Claim 4.5. *Let $C \subseteq \mathbb{F}^N$ be a length $n = |N|$ code with rate ρ, and let $C \subseteq \mathbb{F}^{[b]}$ be a code with rate ρ_b. Let $\mathcal{P} = \{P_i \mid i \in N\}$ be a set of partitions of N into parts of size b. Then $(C)^2_{\mathcal{P},C_b}$ is a code of length n^2 over $\mathbb{F}$ having rate at least $\rho + \rho_b - 1$.*

Proof. That the length of $(C)^2_{\mathcal{P},C_b}$ is n^2 and that it is over $\mathbb{F}$ is immediate from the definition. As for the rate, by inspecting the subspace orthogonal to $(C)^2_{\mathcal{P},C_b}$, it can be seen that the dimension of $((C)^2_{\mathcal{P},C_b})^\perp$ is at most

$$\ell = n \cdot (1-\rho)n + n \cdot \frac{n}{b}(1-\rho_b)b.$$

From that we conclude the rate of $(C)^2_{\mathcal{P},C_b}$ is at least $1 - \frac{\ell}{n^2} = \rho + \rho_b - 1$. □

We now show that if the partition sequence used is row-evasive, the squaring operation preserves the RLCC property.

Proposition 4.6. *Let $C \subseteq \mathbb{F}^N$ be a (q, δ, ε)-RLCC of length $n = |N|$, and let $C_b \subseteq \mathbb{F}^{[b]}$ be a $(q_b, \delta_b, \varepsilon_b)$-RLCC. Further let $\mathcal{P} = \{P_i \mid i \in N\}$ be a set of partitions of N into parts of size b which is row-evasive. Then, $C' = (C)^2_{\mathcal{P},C_b}$ is a $(q', \delta', \varepsilon')$-RLCC for*

$$q' = 64(q+q_b)\frac{1}{\delta\delta_b},$$
$$\delta' = \frac{\delta^2\delta_b}{8},$$
$$\varepsilon' = \frac{5}{9} + \frac{4}{9}\max(\varepsilon_b, \varepsilon).$$

Proof. To show that C' is a $(q', \delta', \varepsilon')$-RLCC we need to devise a corrector Cor' for it with the desired parameters. Towards that, let Cor be a corrector for C. For every $j \in N$ and $B \in P_j$, let Cor_B be a corrector for C_B.

The corrector. For $i, j \in N$, $\mathsf{Cor}'(i, j)$ with oracle access to $z \in \mathbb{F}^{N\times N}$ proceeds as follows.

1. Invoke $\mathsf{Cor}(j)$ with oracle access to $z_{\{i\}\times N}$, and denote the output by $\sigma \in \mathbb{F} \cup \{\perp\}$. If $\sigma = \perp$, halt and output $\perp$.

2. Sample uniformly and independently at random $m = \frac{4}{\delta}$ indices $j_1, \dots, j_m \in N$. For every $t \in [m]$ query z_{i,j_t}.

3. For every $t \in [m]$ proceed as follows: Invoke $\mathsf{Cor}_{[P_{j_t}]_i}(i)$ with oracle access to $z_{[P_{j_t}]_i}$ to obtain an output $\sigma_t \in \mathbb{F} \cup \{\perp\}$. If for some $t \in [m]$ either $\sigma_t = \perp$ or $\sigma_t \neq z_{i,j_t}$, halt and output $\perp$.

4. For every $t \in [m]$ sample $d = \frac{4}{\delta_b}$ indices $i_t^1, \dots, i_t^d \in (P_{j_t})_i$ uniformly and independently at random, and query $z_{i_t^r, j_t}$ for each $r \in [d]$.

5. For every $t \in [m]$ and $r \in [d]$ invoke $\mathsf{Cor}(j_t)$ with oracle access to $z_{\{i_t^r\} \times N}$, and set $\sigma_t^r \in \mathbb{F} \cup \{\perp\}$ to be the resulted output. If for some $t \in [m], r \in [d]$ either $\sigma_t^r = \perp$ or $\sigma_t^r \neq z_{i_t^r, j_t}$, halt and output $\perp$.

6. Output σ.

Correctness. That $\mathsf{Cor}'(i,j)$ always outputs $c_{i,j}$, when given oracle access to $c \in C'$, is immediate from the fact that Cor and the correctors $\{\mathsf{Cor}_B \mid j \in N, B \in P_j\}$ always correctly output the desired symbol when given access to codewords of their respective codes.

Query analysis. It can be seen that the number of queries that $\mathsf{Cor}'(i,j)$ makes satisfies

$$\begin{aligned} q' &\leqslant q + m + mq_b + md + mdq \\ &\leqslant 4md(q + q_b) \\ &= 64(q + q_b)\frac{1}{\delta\delta_b}. \end{aligned}$$

Error analysis. It remains to bound the probability that $\mathsf{Cor}'(i,j)$ outputs a wrong symbol in $\mathbb{F}$. Towards that, let $z \in \mathbb{F}^{N \times N}$ be such that $\mathsf{dist}(z,c) \leqslant \delta' n^2$ for some $c \in C'$. Define

$$\mathcal{B} = \left\{ i \in N \mid \mathsf{dist}\left(z_{\{i\} \times N}, c_{\{i\} \times N}\right) > \delta n \right\}.$$

By an averaging argument,

$$|\mathcal{B}| < \frac{\delta'}{\delta} \cdot n. \tag{4.2}$$

We consider two cases. If it is the case that $i \notin \mathcal{B}$, then $\sigma \in \mathbb{F} \cup \{\perp\}$, obtained in Step (1), satisfies $\sigma \in \{c_{i,j}, \perp\}$ with probability at least $1 - \varepsilon$, by the guarantee of Cor. By inspecting the corrector Cor' one can see that it either outputs σ or $\perp$ (namely, it never outputs a field element other than σ). It follows that in this case the output of $\mathsf{Cor}'(i,j)$ is in $\{\perp, c_{i,j}\}$ with probability at least $1 - \varepsilon \geqslant 1 - \varepsilon'$, as required.

Hence, we may proceed under the assumption that $i \in \mathcal{B}$. Define

$$\mathcal{J} = \{ j' \in N \mid z_{i,j'} \neq c_{i,j'} \},$$

and notice that per our assumption that $i \notin \mathcal{B}$, $|\mathcal{J}| > \delta n$. Further define

$$\mathcal{J}' = \left\{ j' \in N \,\middle|\, |(P_{j'})_i \cap \mathcal{B}| > \frac{1}{2}\delta_b b \right\}.$$

In what follows we will argue that given that some index j_t sampled in Step (2) by $\mathsf{Cor}'(i,j)$ satisfies $j_t \in \mathcal{J}\backslash\mathcal{J}'$, $\mathsf{Cor}'(i,j)$ outputs $\perp$ with good probability. Therefore, it is desirable to bound $|\mathcal{J}\backslash\mathcal{J}'|$ from below.

Claim 4.7.

$$|\mathcal{J}\backslash\mathcal{J}'| \geqslant \frac{\delta n}{2}.$$

Proof. Towards that, note that

$$\sum_{j'\in\mathcal{J}'} |(P_{j'})_i \cap \mathcal{B}| > |\mathcal{J}'|\frac{1}{2}\delta_b b. \tag{4.3}$$

On the other hand, counting by columns rather than by rows, we get

$$\begin{aligned}\sum_{j'\in\mathcal{J}'} |(P_{j'})_i \cap \mathcal{B}| &= \sum_{j'\in\mathcal{J}'}\sum_{i'\in\mathcal{B}\backslash\{i\}} |(P_{j'})_i \cap \{i'\}| \\ &= \sum_{i'\in\mathcal{B}\backslash\{i\}}\sum_{j'\in\mathcal{J}'} |(P_{j'})_i \cap \{i'\}| \\ &= \sum_{i'\in\mathcal{B}\backslash\{i\}} |\{j'\in\mathcal{J}' \mid i' \in (P_{j'})_i\}|\end{aligned}$$

Therefore,

$$\begin{aligned}\sum_{j'\in\mathcal{J}'} |(P_{j'})_i \cap \mathcal{B}| &\leqslant \sum_{i'\in\mathcal{B}\backslash\{i\}} |\{j'\in N \mid i' \in (P_{j'})_i\}| \\ &\leqslant (|\mathcal{B}|-1)\cdot 2b, \end{aligned} \tag{4.4}$$

where the last inequality follows from by the fact that $\mathcal{P}$ is row-evasive. Equations (4.2) to (4.4) imply

$$|\mathcal{J}'| < \frac{4|\mathcal{B}|}{\delta_b} < 4\frac{\delta'}{\delta\delta_b}n.$$

By our choice of δ' we conclude that

$$|\mathcal{J}'| \leqslant \frac{\delta n}{2},$$

and so, as $i \in \mathcal{B}$,

$$|\mathcal{J}\backslash\mathcal{J}'| \geqslant \frac{\delta n}{2}.$$

$\square$

By Claim 4.7 and since $\mathsf{Cor}'(i,j)$ samples $m = \frac{4}{\delta}$ indices $j_t \in N$, uniformly and independently at random, with probability at least $1 - \frac{1}{e^2} > \frac{2}{3}$, it holds that for some $t \in [m]$,

$j_t \in \mathcal{J} \backslash \mathcal{J}'$. Let E_A denote this event. We therefore condition on that the event E_A occurred, and proceed by bounding the (conditioned) probability that the output of Cor' is "legal", namely, the output is in $\{\sigma, \perp\}$.

Let $t \in [m]$ be an element such that $j_t \in \mathcal{J} \backslash \mathcal{J}'$. We consider two cases. If

$$\mathsf{dist}\left(z_{[P_{j_t}]_i \times \{j_t\}}, c_{[P_{j_t}]_i \times \{j_t\}}\right) \leqslant \delta_b b, \tag{4.5}$$

then with probability at least $1 - \varepsilon_b$, $\sigma_t \in \{\perp, c_{i,j_t}\}$. If $\sigma_t = \perp$ then the output of Cor' is legal. Otherwise, as $j_t \in \mathcal{J}$, we have that $\sigma_t = c_{i,j_t} \neq z_{i,j_t}$ and so in Step (3), Cor' will output $\perp$ which is, again, a legal output. We conclude that if Equation (4.5) holds then $\mathsf{Cor}'(i,j)$ outputs $\perp$ with probability at least $1 - \varepsilon_b$.

Consider then the case where Equation (4.5) does not hold, and let

$$\mathcal{I} = \left\{ i' \in (P_{j_t})_i \mid z_{i',j_t} \neq c_{i',j_t} \right\}.$$

Per our assumption (and since $z_{i,j_t} \neq c_{i,j_t}$), we have that $|\mathcal{I}| > \delta_b b - 1$. Therefore $|\mathcal{I}| \geqslant \delta_b b$.[3] Consider the set $\mathcal{I} \backslash \mathcal{B} = \mathcal{I} \backslash ((P_{j_t})_i \cap \mathcal{B})$. As $j_t \notin \mathcal{J}'$, we have that $|(P_{j_t})_i \cap \mathcal{B}| \leqslant \frac{1}{2}\delta_b b$. Therefore,

$$\begin{aligned} |\mathcal{I} \backslash \mathcal{B}| &\geqslant |\mathcal{I}| - |(P_{j_t})_i \cap \mathcal{B}| \\ &\geqslant \delta_b b - \frac{1}{2}\delta_b b \\ &= \frac{1}{2}\delta_b b. \end{aligned}$$

Hence, as $\mathsf{Cor}'(i,j)$ samples $d = \frac{4}{\delta_b}$ indices $i_t^1, \ldots, i_t^d \in (P_{j_t})_i$ uniformly and independently at random, with probability at least $1 - \frac{1}{e^2} > \frac{2}{3}$, for some $r \in [d]$, it occurs that $i_t^r \in \mathcal{I} \backslash \mathcal{B}$. Let E_B denote this event. We therefore further condition on that E_B occurred (recall, we already conditioned on the event E_A).

Let $r \in [d]$ be an element such that $i_t^r \in \mathcal{I} \backslash \mathcal{B}$. As $i_t^r \notin \mathcal{B}$, we have that with probability at least $1 - \varepsilon$ over the randomness of $\mathsf{Cor}(j_t)$ invoked in Step (5) with oracle access to $z_{\{i_t^r\} \times N}$, it holds that $\sigma_t^r \in \{c_{i_t^r, j_t}, \perp\}$. Since we also know that $z_{i_t^r, j_t} \neq c_{i_t^r, j_t}$ (as $i_t^r \in \mathcal{I}$), it follows that with probability at least $1 - \varepsilon$, $\sigma_t^r \neq z_{i_t^r, j_t}$, and therefore $\mathsf{Cor}'(i,j)$ outputs $\perp$ with probability at least $1 - \varepsilon$ in the case that Equation (4.5) does not hold.

[3]We assume, without loss of generality, that δ_b is taken to be the maximal value such that C_b is a $(q_b, \delta_b, \varepsilon_b)$-RLCC. This implies, in particular, that $\delta_b b$ is an integer. We may make this assumption since plugging in a smaller value of δ_b in the proposition results in a weaker statement.

To conclude the error analysis, we have that if $i \in \mathcal{B}$ then

$$\begin{aligned}
\mathbf{Pr}[\mathsf{Cor}'(i,j) \in \{c_{i,j}, \perp\}] &\geqslant \mathbf{Pr}[E_A] \cdot \mathbf{Pr}[\mathsf{Cor}'(i,j) \in \{c_{i,j}, \perp\} \mid E_A] \\
&\geqslant \frac{2}{3} \cdot \mathbf{Pr}[\mathsf{Cor}'(i,j) \in \{c_{i,j}, \perp\} \mid E_A] \\
&\geqslant \frac{2}{3} \cdot \min(1-\varepsilon_b, \mathbf{Pr}[E_B \mid E_A] \cdot \mathbf{Pr}[\mathsf{Cor}'(i,j) \in \{c_{i,j}, \perp\} \mid E_A \cap E_B]), \\
&\geqslant \frac{2}{3} \cdot \min(1-\varepsilon_b, \frac{2}{3} \cdot \mathbf{Pr}[\mathsf{Cor}'(i,j) \in \{c_{i,j}, \perp\} \mid E_A \cap E_B]), \\
&\geqslant \frac{2}{3} \cdot \min(1-\varepsilon_b, \frac{2}{3} \cdot (1-\varepsilon)),
\end{aligned}$$

where the minimum in the third inequality is due to the two possibilities of whether or not Equation (4.5) holds, and the last inequality follows by the discussion above. Therefore,

$$\begin{aligned}
\mathbf{Pr}[\mathsf{Cor}'(i,j) \in \{c_{i,j}, \perp\}] &\geqslant \min(1-\varepsilon, \frac{2}{3} \cdot \min(1-\varepsilon_b, \frac{2}{3} \cdot (1-\varepsilon))), \\
&= \frac{2}{3} \cdot \min(1-\varepsilon_b, \frac{2}{3} \cdot (1-\varepsilon)) \\
&\geqslant \frac{4}{9} \cdot \min(1-\varepsilon_b, 1-\varepsilon) \\
&= \frac{4}{9} - \frac{4}{9} \max(\varepsilon_b, \varepsilon) \\
&= 1 - \varepsilon',
\end{aligned}$$

where the first minimum is for considering the two possibilities of whether $i \in \mathcal{B}$ (recall that the case that $i \notin \mathcal{B}$ was discussed at the start of the analysis). This shows that the error probability is as stated, and the proposition follows. □

5 Deriving the main result

In this section we prove our main result, Theorem 1.1. To this end, in Section 5.1 we devise a distance amplification procedure (or, more accurately, a correction-radius amplification procedure) for RLCC. This procedure is quite similar to previously known procedures [AEL95, AL96, KMRS17]. However, there is some technical nuances that make it difficult for us to invoke those in a black-box manner. Further, we give a somewhat more simplified construction and analysis, suitable to our needs. Finally, in Section 5.2, we prove Theorem 1.1.

5.1 Distance amplification for RLCC

We start by defining *relaxed-local-amplifiers.*

Definition 5.1. *Let N be a finite set such that $|N| = n$ and let $\mathbb{F}$ be a field. A linear subspace $L \subseteq \mathbb{F}^N$ is called a (q, δ, α)-*relaxed-local-amplifier *if there exists a deterministic procedure* $\mathsf{Cor} : N \to \mathbb{F}$ *that is given an oracle access to $z \in \mathbb{F}^N$ and has the following guarantees:*

1. *For every $i \in N$ and $y \in L$,* $\mathsf{Cor}(i) = y_i$.
2. *For every $i \in N$, $c \in L$ and $z \in \mathbb{F}^N$ such that* $\mathsf{dist}(z, c) \leqslant \delta n$, *it holds that* $\mathsf{Cor}(i) \in \{c_i, \perp\}$ *for at least $(1-\alpha)$-fraction of the indices $i \in N$.*
3. Cor *always makes at most q queries to z.*

Relaxed-local-amplifiers give rise to the following claim.

Claim 5.2. *Let $C \subseteq \mathbb{F}^N$ be a (q, δ, ε)-RLCC and let $L \subseteq \mathbb{F}^N$ be a (q', δ^*, α)-relaxed-local-amplifier such that $\alpha = \delta$. Then, $\bar{C} = C \cap L$ is a $(qq', \delta^*, \varepsilon)$-RLCC.*

Proof. The proof goes by demonstrating a corrector $\overline{\mathsf{Cor}}$ for $\bar{C}$ which attains the stated parameters. Towards that, let $n = |N|$, and let Cor be a corrector for C, and let Cor_L be a corrector for L. For $c \in \bar{C}$ and $z \in \mathbb{F}^N$, define $z^c \in \mathbb{F}^N$ to be the string that is given by

$$\forall j \in N \quad z^c_j = \begin{cases} \mathsf{Cor}^z_L(j) & \mathsf{Cor}^z_L(j) \neq \perp \ ; \\ c_j & \mathsf{Cor}^z_L(j) = \perp \ . \end{cases}$$

Note that z^c is a well-defined, fixed, element in $\mathbb{F}^N$ as Cor^z_L is deterministic. By the guarantee of Cor_L, for every $c \in \bar{C}$ and $z \in \mathbb{F}^N$ such that $\mathsf{dist}(z, c) \leqslant \delta^* n$, we have that

$$\mathsf{dist}(z^c, c) \leqslant \alpha n = \delta n. \tag{5.1}$$

We turn to define the corrector $\overline{\mathsf{Cor}}$ for $\bar{C}$. For $z \in \mathbb{F}^N$ and $i \in N$, $\overline{\mathsf{Cor}}^z(i)$ proceeds as follows:

1. Simulate $\mathsf{Cor}(i)$ but, in the process, for every query index $j \in N$ that $\mathsf{Cor}^z(i)$ asks to query:
   - (a) Run $\mathsf{Cor}^z_L(j)$ in attempt to obtain z^c_j (if $\mathsf{Cor}^z_L(j) \neq \perp$, z^c_j is successfully obtained).
   - (b) If the attempt failed, namely, $\mathsf{Cor}^z_L(j) = \perp$, halt and output $\perp$.
   - (c) Feed $\mathsf{Cor}(i)$ with z^c_j.
2. Return the result returned by $\mathsf{Cor}(i)$.

It is immediate that the number of queries that $\overline{\mathsf{Cor}}$ makes is at most qq'. As for the correctness, first, it can be easily verified that in the case that $\overline{\mathsf{Cor}}(i)$ is given access to a codeword $c \in \overline{\mathsf{Cor}}$, it outputs c_i (this follows by the respective guarantees of Cor and Cor_L). Secondly, we turn to analyze the case that $\overline{\mathsf{Cor}}(i)$ is given access to a possibly corrupted codeword, and so assume that $z \in \mathbb{F}^N$ is such that $\mathsf{dist}(z, c) \leqslant \delta^* n$ for some $c \in \bar{C}$.

Consider a run of $\overline{\mathsf{Cor}}^z(i)$, and let A denote the event that during the simulation of $\mathsf{Cor}(i)$, an index $j \in N$ such that $\mathsf{Cor}_L^z(j) = \perp$ is requested. Further consider as a thought experiment a hypothetical run, this time of $\mathsf{Cor}^{z^c}(i)$, and let B denote the event that an index $j \in N$ such that $\mathsf{Cor}_L^z(j) = \perp$ is queried. As during the run of $\overline{\mathsf{Cor}}^z(i)$, until the first time that the simulated $\mathsf{Cor}(i)$ requests $j \in N$ such that $\mathsf{Cor}_L^z(j) = \perp$, it is fed with symbols identical to those of z^c, we have that $\mathbf{Pr}[A] = \mathbf{Pr}[B]$. Moreover, from the same reason, we have that

$$\mathbf{Pr}\left[\overline{\mathsf{Cor}}^z(i) \in \{c_i, \perp\} \mid \neg A\right] = \mathbf{Pr}\left[\mathsf{Cor}^{z^c}(i) \in \{c_i, \perp\} \mid \neg B\right].$$

Thus,

$$\begin{aligned}
\mathbf{Pr}\left[\overline{\mathsf{Cor}}^z(i) \in \{c_i, \perp\}\right] &= \mathbf{Pr}[A]\,\mathbf{Pr}\left[\overline{\mathsf{Cor}}^z(i) \in \{c_i, \perp\} \mid A\right] \\
&\quad + \mathbf{Pr}[\neg A]\,\mathbf{Pr}\left[\overline{\mathsf{Cor}}^z(i) \in \{c_i, \perp\} \mid \neg A\right] \\
&= \mathbf{Pr}[A] \cdot 1 + \mathbf{Pr}[\neg A]\,\mathbf{Pr}\left[\overline{\mathsf{Cor}}^z(i) \in \{c_i, \perp\} \mid \neg A\right] \\
&= \mathbf{Pr}[B] \cdot 1 + \mathbf{Pr}[\neg B]\,\mathbf{Pr}\left[\mathsf{Cor}^{z^c}(i) \in \{c_i, \perp\} \mid \neg B\right] \\
&\geqslant \mathbf{Pr}[B]\,\mathbf{Pr}\left[\mathsf{Cor}^{z^c}(i) \in \{c_i, \perp\} \mid B\right] \\
&\quad + \mathbf{Pr}[\neg B]\,\mathbf{Pr}\left[\mathsf{Cor}^{z^c}(i) \in \{c_i, \perp\} \mid \neg B\right] \\
&= \mathbf{Pr}\left[\mathsf{Cor}^{z^c}(i) \in \{c_i, \perp\}\right] \\
&\geqslant 1 - \varepsilon,
\end{aligned}$$

where the second equality follows by the fact that in the event A, $\overline{\mathsf{Cor}}^z(i)$ outputs $\perp$. The last inequality follows from the guarantee of Cor, together with Equation (5.1). We thus have that as wanted, given that $\mathsf{dist}(z, c) \leqslant \delta^* n$, $\overline{\mathsf{Cor}}^z(i)$ outputs a correct symbol (in $\{c_i, \perp\}$) with a probability at least $1 - \varepsilon$, which concludes the correctness analysis of $\overline{\mathsf{Cor}}$. □

Relaxed-local-amplifiers exist, and can be found efficiently, as given by the following lemma. The proof for the lemma is by a construction, and this construction resembles that of the a AEL distance amplification [AEL95, AL96] which was adapted and shown to preserve locality features by [KMRS17].

Lemma 5.3. *There exist universal constants η_0 and e_0 such that the following holds. For every set N such that $|N| = n \geqslant \eta_0$, $\mathbb{F}$ a finite field, and δ, α such that $0 \leqslant \delta \leqslant \frac{1}{2}(1 - \frac{1}{|\mathbb{F}|})^2$,*

$0 \leqslant \alpha \leqslant 1$, *there exists a linear subspace* $L \subseteq \mathbb{F}^N$ *which is a* (q, δ, α)*-local-amplifier for* $q \leqslant \left(\frac{2}{\delta\alpha}\right)^{e_0}$, *such that* $\dim L \geqslant (1 - 10\mathrm{H}_{|\mathbb{F}|}(\sqrt{\delta}))n$. *Furthermore, there exists an algorithm that takes as input* n, δ, α, *and computes* L, *that runs time* $|\mathbb{F}|^{\mathrm{poly}\left(\frac{1}{\delta\alpha}\right)} + \mathrm{poly}(n)$.

A proof for the lemma is deferred to Section 5.3.

5.2 The main theorem

We are now ready to prove our main result, Theorem 1.1. First, we give a more formal and precise statement.

Theorem 5.4. *Let* $\mathbb{F}$ *be a field of constant size.*[4] *For every* $\ell \in \mathbb{N}$ *and* $0 \leqslant \delta \leqslant \frac{1}{2}(1 - \frac{1}{|\mathbb{F}|})^2$ *there exists a rate-*ρ $(q, \delta, \frac{1}{3})$*-RLCC* $C \subseteq \mathbb{F}^n$ *of length* $n \in [\ell, \ell^2]$ *where*

$$\begin{aligned} q &= (\log n)^{O(\log\log\log n)}, \\ \rho &= 1 - 10\mathrm{H}_{|\mathbb{F}|}(\sqrt{\delta}) - o(1). \end{aligned}$$

Moreover, there exists an algorithm that gets as input δ *and* ℓ *and computes* C, *as well as an efficient local corrector for* C, *in time* $\mathrm{poly}(\ell)$.

Proof. Let ℓ_0 be a sufficiently large universal constant. We assume without loss of generality that $\ell \geqslant \ell_0$ [5]. We prove the theorem by constructing a sequence of codes $C_0, \ldots, C_m$ iteratively, and the desired code C will be constructed from the code C_m. We start by defining a few parameters. Set

$$\begin{aligned} b &= 2^{\lceil 2\log\log\log \ell \rceil}, \\ \delta^* &= \frac{1}{(\log\log \ell)^4}, \\ n_0 &= 2^{\lceil 2\log\log\log \ell \rceil}, \\ m &= \lceil \log\log \ell - \log\log n_0 \rceil, \\ n &= {n_0}^{2^m}. \end{aligned}$$

Notice that we have that

$$(\log\log \ell)^2 \leqslant b = n_0 \leqslant 2(\log\log \ell)^2, \tag{5.2}$$

[4]Taking a larger than constant field size only affects the running-time part of the theorem, as the time to compute C is, more precisely, $|\mathbb{F}|^{\mathrm{poly}(\log\log n)} + \mathrm{poly}(n)$. Nonetheless, simple adaptations to the construction can be made in cases where a larger field size is needed.

[5]Every $\ell < \ell_0$ trivially satisfies the statement, as can be seen by noting that the identity code $C_{\mathrm{I}} \subseteq \mathbb{F}^{[\ell]}$ is a RLCC with query complexity $\ell = O(1)$ and correction radius $\frac{1}{\ell} = \Omega(1)$.

and

$$\ell = {n_0}^{2^{\log\log \ell - \log\log n_0}} \leqslant n \leqslant {n_0}^{2^{\log\log \ell - \log\log n_0 + 1}} = \ell^2. \tag{5.3}$$

Further, set

$$C_b = \left\{ v \in \mathbb{F}^{[b]} \;\middle|\; \sum_{i \in [b]} v_i = 0 \right\} \subseteq \mathbb{F}^{[b]}.$$

Clearly C_b has rate $\rho_b = 1 - \frac{1}{b}$ and is a $(q_b, \delta_b, \varepsilon_b)$-RLCC for $q_b = b$, $\delta_b = \frac{1}{b}$ and $\varepsilon_b = 0$. We will maintain the invariant that at each step, namely, for every $t \in [m]$, the code C_t is of length $n_0^{2^t}$ and is a $(q_t, \delta^*, \frac{1}{3})$-RLCC, and so, by Equation (5.3), C_m will be of a desired length.

The base code. Set $N_0 = [n_0]$ and take $C_0 \subseteq \mathbb{F}^{N_0}$ to be the identity code. Set $q_0 = n_0$. By Equation (5.2), C_0 is trivially a $(q_0, \delta^*, \frac{1}{3})$-RLCC, (since $\delta^* < \frac{1}{n_0}$ and assuming we take $\ell_0 \geqslant 7$), with rate $\rho_0 = 1$.

The iterative step. For $t = 1, \ldots, m$, C_t is constructed as follows.

1. Set $N_t = N_{t-1} \times N_{t-1}$ and $n_t = |N_t|$, and note that $n_t = n_{t-1}^2$.
2. Set $\mathcal{P}_t = \{P_i \mid i \in N_{t-1}\}$ to be a set of partitions of N_{t-1} into parts of size b that is row-evasive. Such $\mathcal{P}_t$ exists by Claim 4.2, and note that the claim is applicable as both b and n_{t-1} are powers of 2, and $b = n_0 \leqslant n_{t-1}$.
3. Set $C_t' = (C_{t-1})^2_{\mathcal{P}_t, C_b}$. By Claim 4.5, C_t' has rate at least

$$\rho_t' = \rho_{t-1} - \frac{1}{b}. \tag{5.4}$$

By Proposition 4.6, C_t' is a $(q_t', \delta_t', \varepsilon_t')$-RLCC for

$$\begin{aligned} q_t' &= 2^6(q_{t-1} + b)b\frac{1}{\delta^*} \leqslant 2^7 q_{t-1} b \frac{1}{\delta^*} \leqslant 2^8 (\log\log \ell)^6 q_{t-1}, \\ \delta_t' &= \frac{1}{8}(\delta^*)^2 \frac{1}{b} \geqslant \frac{1}{16(\log\log \ell)^{10}}, \\ \varepsilon_t' &= \frac{5}{9} + \frac{4}{9} \cdot \frac{1}{3} = \frac{19}{27}, \end{aligned}$$

where, for these inequalities, we used Equation (5.2) and the fact that $q_{t-1} \geqslant q_0 \geqslant b$. Moreover, by Claim 3.7, C_t' is a $(q_t'', \delta_t', \frac{1}{3})$-RLCC for

$$q_t'' = 4q_t' \leqslant 2^{10} (\log\log \ell)^6 q_{t-1},$$

as $\left(\frac{19}{27}\right)^4 < \frac{1}{3}$.

4. Set $L_t \subseteq \mathbb{F}^{N_t}$ to be a linear subspace which is a $(q_{L_t}, \delta_{L_t}, \alpha_{L_t})$-local-amplifier for $\delta_{L_t} = \delta^*$, $\alpha_{L_t} = \delta'_t$. By Lemma 5.3 such a subspace exists with

$$q_{L_t} \leqslant \left(\frac{2}{\delta^* \delta'_t}\right)^{e_0} \leqslant \left(\frac{32(\log\log \ell)^{10}}{\delta^*}\right)^{e_0} = 2^{5e_0}(\log\log \ell)^{14e_0},$$

where e_0 is the constant from Lemma 5.3. Moreover, L_t satisfies

$$\dim(L_t) \geqslant (1 - 10\mathrm{H}_{|\mathbb{F}|}(\sqrt{\delta^*}))n_t.$$

Note that the lemma is applicable since its prerequisites are met. Indeed, we have that $n_t \geqslant \eta_0$, the universal constant from the lemma, since $n_t \geqslant n_0 \geqslant (\log\log \ell_0)^2$ as long as we choose ℓ_0 such that $(\log\log \ell_0)^2 \geqslant \eta_0$. Moreover, the requirement regarding δ_{L_t} is satisfied as well as, assuming that $\ell_0 \geqslant 16$, we have that

$$\delta_{L_t} = \delta^* = \frac{1}{(\log\log \ell)^4} \leqslant \frac{1}{(\log\log \ell_0)^4} \leqslant \frac{1}{16} \leqslant \frac{1}{2}\left(1 - \frac{1}{|\mathbb{F}|}\right)^2,$$

since $|\mathbb{F}| \geqslant 2$.

5. Set $C_t = C'_t \cap L_t$. By Claim 3.5, the rate of C_t is at least

$$\rho_t = \rho'_t + \frac{\dim(L_t)}{n_t} - 1 \geqslant \rho_{t-1} - \frac{1}{b} - 10\mathrm{H}_{|\mathbb{F}|}(\sqrt{\delta^*}), \tag{5.5}$$

where we used Equation (5.4). By Claim 5.2, C_t is a $(q_t, \delta^*, \frac{1}{3})$-RLCC for

$$q_t = q''_t q_{L_t} \leqslant 2^{10+5e_0}(\log\log \ell)^{6+14e_0} q_{t-1}. \tag{5.6}$$

The final code. The code C_m is, as argued, a $(q_m, \delta^*, \frac{1}{3})$-RLCC, of length $n_m = {n_0}^{2^m} = n$. If $\delta \leqslant \delta^*$, we set $C = C_m$ to be the desired code. If otherwise, we set $L \subseteq \mathbb{F}^{N_m}$ to be a (q_L, δ, δ^*)-relaxed-local-amplifier, and by Lemma 5.3, such L exists with query complexity

$$q_L \leqslant \left(\frac{2}{\delta\delta^*}\right)^{e_0} \leqslant \left(\frac{2}{\delta^*}\right)^{2e_0} = 2^{2e_0}(\log\log \ell)^{8e_0}, \tag{5.7}$$

and

$$\dim L \geqslant (1 - 10\mathrm{H}_{|\mathbb{F}|}(\sqrt{\delta}))n.$$

Set $C = C_m \cap L$. In either case, by Claim 5.2, C is a $(q, \delta, \frac{1}{3})$-RLCC for $q = q_m q_L$. By Equation (5.5) and Claim 3.5, C has rate ρ which satisfies

$$\begin{aligned}
\rho &\geqslant \rho_m - 10\mathrm{H}_{|\mathbb{F}|}(\sqrt{\delta}) \\
&\geqslant \rho_0 - \frac{m}{b} - 10m\mathrm{H}_{|\mathbb{F}|}(\sqrt{\delta^*}) - 10\mathrm{H}_{|\mathbb{F}|}(\sqrt{\delta}) \\
&= 1 - \frac{m}{b} - 10m\mathrm{H}_{|\mathbb{F}|}\left(\frac{1}{(\log\log\ell)^2}\right) - 10\mathrm{H}_{|\mathbb{F}|}(\sqrt{\delta}) \\
&\geqslant 1 - \frac{1}{\log\log\ell} - 10(\log\log\ell)\mathrm{H}_{|\mathbb{F}|}\left(\frac{1}{(\log\log\ell)^2}\right) - 10\mathrm{H}_{|\mathbb{F}|}(\sqrt{\delta}) \\
&= 1 - 10\mathrm{H}_{|\mathbb{F}|}(\sqrt{\delta}) - o(1),
\end{aligned}$$

where the last equality holds by Fact 3.1. As for the query complexity, by Equation (5.6) and Equation (5.7), we have that

$$\begin{aligned}
q &= q_m \cdot q_L \\
&\leqslant (2^{10+5e_0}(\log\log\ell)^{6+14e_0})^m q_0 \cdot 2^{2e_0}(\log\log\ell)^{8e_0} \\
&= (\log\log\ell)^{O(\log\log\ell)} \\
&= (\log\ell)^{O(\log\log\ell)} \\
&= (\log n)^{O(\log\log n)},
\end{aligned}$$

where in the third equality we used that $q_0 \leqslant 2(\log\log\ell)^2$ and $m \leqslant \log\log\ell$, and the last equality holds as $n \in [\ell, \ell^2]$ by Equation (5.3). $\square$

Explicitness. It remains to verify that there exists an efficient algorithm that computes C and an efficient local corrector for C. The code C is constructed by taking $C = C_m \cap L$ (or $C = C_m$ if $\delta < \delta^*$). By Lemma 5.3, there exists an algorithm with running-time

$$|\mathbb{F}|^{\mathrm{poly}(\frac{1}{\delta^*})} + \mathrm{poly}(n) = |\mathbb{F}|^{\mathrm{poly}(\log\log n)} + \mathrm{poly}(n) = \mathrm{poly}(n)$$

that computes L. Thus, to show that there exists an algorithm that computes C (namely, a parity-check matrix for C) in polynomial time, it suffices to show that C_m can be computed efficiently. Recall that the base code C_0 is the identity code, and the code C_b used through the construction is a simple parity code - both of which, clearly, can be computed efficiently. For every $t \in [m]$, the code C_t is constructed by computing $\mathcal{P}_t$, $(C_{t-1})^2_{\mathcal{P}_t, C_b}$ and L_t, and by performing few other minor operations that are clearly done in time $\mathrm{poly}(n_t)$. The row-evasive sets of partitions $\mathcal{P}_t$ are achieved by Claim 4.2, and the claim's proof contains an explicit construction which clearly can be implemented in time $\mathrm{poly}(n_t)$. The relaxed-local-amplifier L_t, by Lemma 5.3, can be computed in time

$$|\mathbb{F}|^{\mathrm{poly}\left(\frac{1}{\delta^*\delta'_t}\right)} + \mathrm{poly}(n_t) = |\mathbb{F}|^{\mathrm{poly}(\log\log n)} + \mathrm{poly}(n_t).$$

It can be readily verified, by Definition 4.4, that there exists an algorithm, that given the result of the computation of C_{t-1} and C_b, and given $\mathcal{P}_t$, computes $(C_{t-1})^2_{\mathcal{P}_t,C_b}$, and runs in time $\mathrm{poly}(n_t)$. Therefore, in total, the time it takes to compute C_t given C_{t-1} is $|\mathbb{F}|^{\mathrm{poly}(\log\log n)} + \mathrm{poly}(n_t)$. Thus, the sequence $C_1, \ldots, C_m$ can be computed in time $m(|\mathbb{F}|^{\mathrm{poly}(\log\log n)} + \mathrm{poly}(n)) = \mathrm{poly}(n)$. We conclude that there exists an algorithm that computes C of the stated properties, which runs in time $\mathrm{poly}(n)$. That there is an efficient corrector for C readily follows by inspecting the construction.

5.3 Proof for Lemma 5.3

In this section we provide a proof for Lemma 5.3. We require some preliminaries.

Theorem 5.5 (The Gilbert-Varshamov bound, [Gil52, Var57])**.** *For any $n \in \mathbb{N}$, a field $\mathbb{F}$ of size q, and $0 \leqslant \delta \leqslant 1 - \frac{1}{q}$, there exists a linear code $C \subseteq \mathbb{F}^{[n]}$, with relative distance δ and rate $1 - \mathrm{H}_q(\delta) - \frac{2}{n}$.*

Claim 5.6. *For every field $\mathbb{F}$ of size q, there exists an algorithm that takes as input $n \in \mathbb{N}$ and $\delta \leqslant 1 - \frac{1}{q}$, and computes a code $C \subseteq \mathbb{F}^{[n]}$ with distance δ and rate $1 - \mathrm{H}_q(\delta) - \frac{2}{n}$. The running-time of the algorithm is $q^{O(n^2)}$.*[6]

Proof. Since the codes considered by Theorem 5.5 are linear, for each such code $C \subseteq \mathbb{F}^{[n]}$, one can consider the parity-check matrix, $P \in \mathbb{F}^{(n-k)\times n}$, where k is the dimension of C. Recall that the parity-check matrix satisfies $C = \{v \in \mathbb{F}^n \mid Pv = 0\}$. There are $q^{(n-k)n} \leqslant q^{n^2}$ such matrices, and for every matrix P, the distance of the code it induces can be checked simply by going over all of the vectors in the corresponding code and compute the hamming weight. This can be done in time $q^{O(n)}$. $\square$

The proof of Lemma 5.3 relies on special graphs called *samplers*, which we now define. In the definition we will use the following notation. For a bipartite graph $G = (U, V, E)$ and $u \in U$ we denote by $N(u)$ the set $\{v \in V \mid (u, v) \in E\}$.

Definition 5.7. *Let $0 < \beta, \gamma < 1$. A bipartite graph $G = (U, V, E)$ is called an (α, γ)-*sampler *if for every subset $P \subseteq V$, for all but γ-fraction of vertices $u \in U$ it holds that*

$$\left| \frac{|N(u) \cap P|}{|N(u)|} - \frac{|P|}{|V|} \right| < \beta.$$

[6] We remark that in fact such codes can be found in time $|\mathbb{F}|^{O(n)}$ (by performing an exhaustive search on a limited family of matrices, see [GRS12]), but the bound $|\mathbb{F}|^{O(n^2)}$ suffices for the purposes of this work.

We will make use of the following explicit samplers (see [KMRS17] Lemma 2.12 and references therein).

Lemma 5.8 ([KMRS17])**.** *There exist universal constants ℓ_0, d_0 such that the following holds. For every $0 < \beta, \gamma < 1$ and $\ell \geqslant \ell_0$ there exists a bipartite d-regular graph $G_{\ell,\alpha,\gamma} = (U, V, E)$ for $|U| = |V| = \ell$ with*

$$d = d(\beta, \gamma) \leqslant \left(\frac{1}{\beta\gamma}\right)^{d_0},$$

such that $G_{\ell,\beta,\gamma}$ is a (β, γ)-sampler. Furthermore, there exists an algorithm that takes as input n, β, γ, and a vertex w of $G_{\ell,\beta,\gamma}$, and computes the list of its neighbors in the graph in time $\mathrm{poly}\left(\frac{\log \ell}{\beta\gamma}\right)$.

We are now in a position to prove the Lemma 5.3.

Proof of Lemma 5.3. Set $e_0 = 2d_0 \log \ell_0$ and $\eta_0 = \ell_0^2$ where d_0 and ℓ_0 are the universal constants from Lemma 5.8.

The subspace construction. We start by setting up some intermediate objects. Set $\beta = \sqrt{\frac{\delta}{2}}$ and $\gamma = \alpha$. Let $G = G_{\ell,\beta,\gamma} = (U, V, E)$ be a bipartite d-regular graph which is a (β, γ)-sampler, whose existence is guaranteed by Lemma 5.8, for $d = d(\beta, \gamma)$ and $\ell = \frac{n}{d}$. We will assume without loss of generality that $\ell \geqslant \ell_0$, and so Lemma 5.8 is applicable. Since, if otherwise it is the case that $n < d\ell_0$, we can simply take the subspace L to be equal to a code $C \subseteq \mathbb{F}^N$ with distance $\Delta = 2\delta$ and rate at least $1 - \mathrm{H}_{|\mathbb{F}|}(\Delta) - \frac{2}{n}$, whose existence is promised by Theorem 5.5. In such case, C is trivially an $(n, \delta, 0)$-relaxed-local-amplifier, and since $n < d\ell_0$, its query complexity is small enough. It can be readily verified that in this case C possesses all of the claimed properties. Indeed, the rate is at least

$$1 - 2\mathrm{H}_{|\mathbb{F}|}(\delta) - \frac{2}{n} \geqslant 1 - 4\mathrm{H}_{|\mathbb{F}|}(\delta)$$

(assuming without loss of generality that $\delta \geqslant \frac{1}{n}$ and using Fact 3.2), and the query complexity is bounded above by

$$d(\beta, \gamma)\ell_0 \leqslant \left(\frac{1}{\beta\gamma}\right)^{d_0} \ell_0 = \left(\frac{1}{\sqrt{\delta/2}\alpha}\right)^{d_0} \ell_0 \leqslant \left(\frac{2}{\delta\alpha}\right)^{e_0}.$$

Further, there exists an algorithm that computes C and runs time $|\mathbb{F}|^{O(n^2)} = |\mathbb{F}|^{\mathrm{poly}(\frac{1}{\delta\alpha})}$, by Claim 5.6.

Thus, we proceed with the sampler $G = (U, V, E)$ as defined above, under the assumption that $\ell \geqslant \ell_0$. Note that $|E| = n$, and let $f : E \to N$ be an arbitrary fixed bijection.

For every $w \in U \cup V$, denote by D_w the set $\{f(e) \mid e \in E, w \in e\}$ and let $g_w : D_w \to [d]$ be an arbitrary fixed bijection. Further let $C_d \subseteq \mathbb{F}^{[d]}$ be a code of length d, with relative distance

$$\Delta_d = \sqrt{2\delta} \tag{5.8}$$

and rate

$$\rho_d \geqslant 1 - \mathrm{H}_{|\mathbb{F}|}(\Delta_d) - \frac{2}{d}. \tag{5.9}$$

The existence of such a code C_d follows from Theorem 5.5, and notice that per our assumption that $\delta \leqslant \frac{1}{2}(1 - \frac{1}{|\mathbb{F}|})^2$, its perquisite is met. Define for every $w \in U \cup V$ the code $C_w \subseteq \mathbb{F}^{D_w}$ to be $\{c \circ g_w \mid c \in C_d\}$. We argue that the following subspace

$$L = \left\{ y \in \mathbb{F}^N \ \mid \ \forall w \in U \cup V \quad y_{D_w} \in C_w \right\}$$

is of the claimed properties.

Explicitness. By Lemma 5.8, there exists an algorithm that computes the adjacency list of each $w \in U \cup V$ and runs in time $\mathrm{poly}\left(\frac{\log \ell}{\beta\gamma}\right) = \mathrm{poly}(n)$. By Claim 5.6, the exists an algorithm that computes C_d and runs in time $|\mathbb{F}|^{O(d^2)} = |\mathbb{F}|^{\mathrm{poly}\left(\frac{1}{\delta\alpha}\right)}$. Thus, a parity-check matrix for L can be efficiently computed in the following manner. First compute a parity-check matrix for C_d whose rows are $o_1, \ldots, o_{t_d} \in \mathbb{F}^{[d]}$. Then, for every $w \in U \cup V$, compute its adjacency list and from it the set D_w (using the bijection f). Then, construct the functions $o_1 \circ g_w, \ldots, o_{t_d} \circ g_w \in \mathbb{F}^{D_w}$ and expand them to functions $h_1^w, \ldots, h_{t_d}^w$ in $\mathbb{F}^N$ ($h_j^w(i) = 0$ for every $i \notin D_w$). The rows of the desired parity-check matrix for L are given by $\{h_1^w, \ldots, h_{t_d}^w \mid w \in U \cup V\}$, and so we output the set of these functions (which were computed during the iterations).

By the account given regarding the time to compute C_d and each adjacency list, the total running-time is $|\mathbb{F}|^{\mathrm{poly}\left(\frac{1}{\delta\alpha}\right)} + \mathrm{poly}(n)$, since it is straightforward that all other operations, during each iteration, can be performed in time $\mathrm{poly}(n)$ – and there are less than $2n$ iterations.

Dimension analysis. We assume without loss of generality that $\Delta_d \geqslant \frac{1}{d}$, as otherwise we can simply take $C_d = \mathbb{F}^{[d]}$, and get that $\dim L = n$. By Equation (5.9) we have that

$$\begin{aligned} \rho_d &\geqslant 1 - \mathrm{H}_{|\mathbb{F}|}(\Delta_d) - \frac{2}{d} \\ &\geqslant 1 - \mathrm{H}_{|\mathbb{F}|}(\Delta_d) - 2\Delta_d \\ &\geqslant 1 - 3\mathrm{H}_{|\mathbb{F}|}(\Delta_d), \end{aligned}$$

where the last inequality follows by Fact 3.2. Plugging in Equation (5.8), and using Fact 3.3, it follows that

$$\rho_d \geqslant 1 - 3\mathrm{H}_{|\mathbb{F}|}(\sqrt{2\delta}) \geqslant 1 - 5\mathrm{H}_{|\mathbb{F}|}(\sqrt{\delta}).$$

To conclude that its dimension is as stated, notice that L is defined by at most

$$2 \cdot \frac{n}{d} \cdot (1-\rho_d)d$$

constraints, and thus

$$\dim L \geqslant (2\rho_d - 1)n \geqslant (1 - 10\mathrm{H}_{|\mathbb{F}|}(\sqrt{\delta}))n.$$

The corrector. We turn to describe a corrector Cor in order to show that L is a (q, δ, α) relaxed-local-amplifier. On input $i \in N$ and oracle access to $z \in \mathbb{F}^N$, $\mathsf{Cor}(i)$ proceeds as follows:

1. Set $e = (u, v) = f^{-1}(i)$;
2. Query $z_u \triangleq z_{D_u}$; If $z_u \notin C_u$, halt and output $\perp$;
3. For every $v' \in N(u)$ query $z_{v'} \triangleq z_{D_{v'}}$ and if $z_{v'} \notin C_{v'}$, halt and output $\perp$;
4. Output z_i.

Correctness. First, the number of queries that Cor makes is exactly $q = d^2$. Therefore,

$$q = d(\beta,\gamma)^2 \leqslant \left(\frac{1}{\beta\gamma}\right)^{2d_0} = \left(\frac{1}{\sqrt{\delta/2\alpha}}\right)^{2d_0} \leqslant \left(\frac{2}{\delta\alpha}\right)^{e_0}.$$

Secondly, it is immediate that for every $i \in N$, $y \in L$, $\mathsf{Cor}(i) = y_i$, as the every check that Cor makes must follow through in this case, by L's definition. Thirdly, it remains to show that for every $i \in N$ and $y \in L$ and $z \in \mathbb{F}^N$ such that $\mathsf{dist}(z, y) \leqslant \delta n$, $\mathsf{Cor}(i) \in \{y_i, \perp\}$ for at least $(1-\alpha)$-fraction of the indices $i \in N$. Towards that, let $y \in L$ and $z \in \mathbb{F}^N$ such that $\mathsf{dist}(z, y) \leqslant \delta n$. Let

$$\mathcal{B} = \{v \in V \mid \mathsf{dist}(z_{D_v}, y_{D_v}) \geqslant \Delta_d d\}.$$

By an averaging argument, we have that

$$|\mathcal{B}| \leqslant \frac{\delta n}{\Delta_d d} = \frac{\delta}{\Delta_d}|V|.$$

As G is a (β,γ)-sampler, we have that for at least $(1-\gamma)$-fraction of the vertices $u \in U$,

$$|N(u) \cap \mathcal{B}| < d\left(\frac{\delta}{\Delta_d} + \beta\right) = d\Delta_d. \tag{5.10}$$

Let

$$\mathcal{A} = \{u \in U \mid u \text{ satisfies Equation (5.10)}\}.$$

Let $i \in N$ such that $f((u,v)) = i$ and suppose that $z_i \neq y_i$. Then, given that $z_{D_u} \in C_u$, as verified by Step (2), it must be that

$$t \triangleq \mathsf{dist}\left(z_{D_u}, y_{D_u}\right) \geqslant \Delta_d d.$$

Let $v_1,\dots,v_t \in N(u)$ be such that for every $r \in [t]$, $z_{f((u,v_r))} \neq y_{f((u,v_r))}$. Given that for every $r \in [t]$, $z_{D_{v_r}} \in C_{v_r}$, as verified by Step 3 of $\mathsf{Cor}(i)$, it must be that for every $r \in [t]$ $\mathsf{dist}(z_{D_{v_r}}, y_{D_{v_r}}) \geqslant \Delta_d d$, i.e., $v_r \in \mathcal{B}$. Therefore, since u has $t \geqslant \Delta_d d$ neighbors $v_r \in N(u)$ which are in $\mathcal{B}$, $u \notin \mathcal{A}$.

It follows that for every $i \in N$ such that $f((u,v)) = i$ for $u \in \mathcal{A}$, either $z_i = y_i$ or one of the checks that $\mathsf{Cor}(i)$ makes is resulted with $\perp$. In either case, for such i, we have that $\mathsf{Cor}(i) \in \{y_i, \perp\}$. Finally, since

$$|\mathcal{A}| \geqslant (1-\gamma)|U| = (1-\alpha)|U|$$

it follows that $\mathcal{I} = \{i \in N \mid i = f((u,v)) \text{ for } u \in \mathcal{A}\}$ satisfies $|I| \geqslant (1-\alpha)n$, since every $u \in \mathcal{A}$ corresponds to exactly d indices $i \in \mathcal{I}$ and $n = d|U|$. Thus, the fraction of indices $i \in N$ which satisfy $\mathsf{Cor}(i) \in \{y_i, \perp\}$ is at least $1-\alpha$, as required. □

References

[AEL95] Noga Alon, Jeff Edmonds, and Michael Luby. Linear time erasure codes with nearly optimal recovery. In *Proceedings of IEEE 36th Annual Foundations of Computer Science*, pages 512–519. IEEE, 1995.

[AG21] Omar Alrabiah and Venkatesan Guruswami. Visible rank and codes with locality. *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, 2021.

[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.

[AS21] Vahid R Asadi and Igor Shinkar. Relaxed locally correctable codes with improved parameters. In *48th International Colloquium on Automata, Languages, and Programming (ICALP 2021)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2021.

[BGGZ21] Jeremiah Blocki, Venkata Gandikota, Elena Grigorescu, and Samson Zhou. Relaxed locally correctable codes in computationally bounded channels. *IEEE Transactions on Information Theory*, 67(7):4338–4360, 2021.

[BGT16] Arnab Bhattacharyya, Sivakanth Gopi, and Avishay Tal. Lower bounds for 2-query LCCs over large alphabet. *arXiv preprint arXiv:1611.06980*, 2016.

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

[BSGH$^+$06] Eli Ben-Sasson, Oded Goldreich, Prahladh Harsha, Madhu Sudan, and Salil Vadhan. Robust PCPs of proximity, shorter PCPs, and applications to coding. *SIAM Journal on Computing*, 36(4):889–974, 2006.

[CG18] Clément L Canonne and Tom Gur. An adaptivity hierarchy theorem for property testing. *Computational Complexity*, 27(4):671–716, 2018.

[CGS20] Alessandro Chiesa, Tom Gur, and Igor Shinkar. Relaxed locally correctable codes with nearly-linear block length and constant query complexity. In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1395–1411. SIAM, 2020.

[CW79] J Lawrence Carter and Mark N Wegman. Universal classes of hash functions. *Journal of computer and system sciences*, 18(2):143–154, 1979.

[CY21a] Gil Cohen and Tal Yankovitz. Lcc and ldc: Tailor-made distance amplification and a refined separation. In *Electronic Colloquium on Computational Complexity (ECCC)*, number 136, 2021.

[CY21b] Gil Cohen and Tal Yankovitz. Rate amplification and query-efficient distance amplification for linear LCC and LDC. In *36th Computational Complexity Conference (CCC 2021)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2021.

[DEL$^+$21] Irit Dinur, Shai Evra, Ron Livne, Alexander Lubotzky, and Shahar Mozes. Locally testable codes with constant rate, distance, and locality. *arXiv preprint arXiv:2111.04808*, 2021.

[DGGW20] Zeev Dvir, Sivakanth Gopi, Yuzhou Gu, and Avi Wigderson. Spanoids—an abstraction of spanning structures, and a barrier for LCCs. *SIAM Journal on Computing*, 49(3):465–496, 2020.

[DGL21] Marcel Dall'Agnol, Tom Gur, and Oded Lachish. A structural theorem for local algorithms with applications to coding, testing, and privacy. In *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1651–1665. SIAM, 2021.

[DGY11] Zeev Dvir, Parikshit Gopalan, and Sergey Yekhanin. Matching vector codes. *SIAM Journal on Computing*, 40(4):1154–1178, 2011.

[DH13] Irit Dinur and Prahladh Harsha. Composition of low-error 2-query PCPs using decodable PCPs. *SIAM Journal on Computing*, 42(6):2452–2486, 2013.

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

[GG16] Oded Goldreich and Tom Gur. Universal locally testable codes. In *Electron. Colloquium Comput. Complex.*, volume 23, page 42, 2016.

[GG21] Oded Goldreich and Tom Gur. Universal locally verifiable codes and 3-round interactive proofs of proximity for csp. *Theoretical Computer Science*, 878:83–101, 2021.

[GGK19] Oded Goldreich, Tom Gur, and Ilan Komargodski. Strong locally testable codes with relaxed local decoders. *ACM Transactions on Computation Theory (TOCT)*, 11(3):1–38, 2019.

[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 comparison of signalling alphabets. *The Bell system technical journal*, 31(3):504–522, 1952.

[GKO$^+$18] Sivakanth Gopi, Swastik Kopparty, Rafael Oliveira, Noga Ron-Zewi, and Shubhangi Saraf. Locally testable and locally correctable codes approaching the Gilbert-Varshamov bound. *IEEE Transactions on Information Theory*, 64(8):5813–5831, 2018.

[GKS13] Alan Guo, Swastik Kopparty, and Madhu Sudan. New affine-invariant codes from lifting. In *Proceedings of the 4th conference on Innovations in Theoretical Computer Science*, pages 529–540. ACM, 2013.

[GKST02] Oded Goldreich, Howard Karloff, Leonard J Schulman, and Luca Trevisan. Lower bounds for linear locally decodable codes and private information retrieval. In *Proceedings 17th IEEE Annual Conference on Computational Complexity*, pages 175–183. IEEE, 2002.

[GL21] Tom Gur and Oded Lachish. On the power of relaxed local decoding algorithms. *SIAM Journal on Computing*, 50(2):788–813, 2021.

[Gol11] Oded Goldreich. Short locally testable codes and proofs. In *Studies in Complexity and Cryptography. Miscellanea on the Interplay between Randomness and Computation*, pages 333–372. Springer, 2011.

[GR17] Tom Gur and Ron D Rothblum. A hierarchy theorem for interactive proofs of proximity. In *8th Innovations in Theoretical Computer Science Conference (ITCS 2017)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2017.

[GR18] Tom Gur and Ron D Rothblum. Non-interactive proofs of proximity. *computational complexity*, 27(1):99–207, 2018.

[GRR20] Tom Gur, Govind Ramnarayan, and Ron Rothblum. Relaxed locally correctable codes. *Theory of Computing*, 16(1):1–68, 2020.

[GRS12] Venkatesan Guruswami, Atri Rudra, and Madhu Sudan. Essential coding theory. *Draft available at http://www. cse. buffalo. edu/ atri/courses/coding-theory/book*, 2012.

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

[KMRS17] Swastik Kopparty, Or Meir, Noga Ron-Zewi, and Shubhangi Saraf. High-rate locally correctable and locally testable codes with sub-polynomial query complexity. *Journal of the ACM (JACM)*, 64(2):11, 2017.

[KSY14] Swastik Kopparty, Shubhangi Saraf, and Sergey Yekhanin. High-rate codes with sublinear-time decoding. *Journal of the ACM (JACM)*, 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 thirty-second annual ACM symposium on Theory of computing*, pages 80–86, 2000.

[KV10] Tali Kaufman and Michael Viderman. Locally testable vs. locally decodable codes. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, pages 670–682. Springer, 2010.

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

[Lip90] Richard J Lipton. Efficient checking of computations. In *Annual Symposium on Theoretical Aspects of Computer Science*, pages 207–215. Springer, 1990.

[LW19] Ray Li and Mary Wootters. Lifted multiplicity codes and the disjoint repair group property. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2019)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2019.

[Mei14] Or Meir. Locally correctable and testable codes approaching the Singleton bound. In *Electronic Colloquium on Computational Complexity (ECCC)*, volume 21, page 14, 2014.

[MR08] Dana Moshkovitz and Ran Raz. Two-query PCP with subconstant error. *Journal of the ACM (JACM)*, 57(5):1–29, 2008.

[PK21] Pavel Panteleev and Gleb Kalachev. Asymptotically good quantum and locally testable classical LDPC codes. *arXiv preprint arXiv:2111.03654*, 2021.

[RZR20] Noga Ron-Zewi and Ron D Rothblum. Local proofs approaching the witness length. In *2020 IEEE 61st Annual Symposium on Foundations of Computer Science (FOCS)*, pages 846–857. IEEE, 2020.

[Var57] R.R. Varshamov. Estimate of the number of signals in error correcting codes. *Docklady Akad. Nauk, SSSR*, 117:739–741, 1957.

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

[Yek08] Sergey Yekhanin. Towards 3-query locally decodable codes of subexponential length. *Journal of the ACM (JACM)*, 55(1):1–16, 2008.