Electronic Colloquium on Computational Complexity, Revision 1 of Report No. 104 (2017)

Local List Recovery of High-Rate Tensor Codes and Applications

Brett Hemenway* Noga Ron-Zewi† Mary Wootters‡

February 25, 2019

Abstract

We show that the tensor product of a high-rate globally list recoverable code is (approximately) locally list recoverable. List recovery has been a useful building block in the design of list decodable codes, and our motivation is to use the tensor construction as such a building block. In particular, instantiating this construction with known constructions of high-rate globally list recoverable codes, and using appropriate transformations, we obtain the first *capacity-achieving* locally list decodable codes (over a large constant size alphabet), and the first capacity-achieving globally list decodable codes with *nearly linear time* list decoding algorithms. Our techniques are inspired by an approach of Gopalan, Guruswami, and Raghavendra (SIAM Journal on Computing, 2011) for list decoding tensor codes.

1 Introduction

List recovery refers to the problem of decoding error correcting codes from "soft" information. More precisely, an *error-correcting code* is a map $C : \Sigma^k \to \Sigma^n$, which maps length-$k$ *messages* to length-n *codewords.* The *rate* of C is the ratio $\rho := k/n$ which measures the amount of redundancy in the encoding. The *relative distance* δ of the code is the minimum fraction of coordinates on which any pair of distinct codewords differ. In general, it is desirable to construct codes C so that both the rate and the distance is large.

The code C is (α, ℓ, L)-*list recoverable* if for any sequence of lists $S_1, \ldots, S_n \subset \Sigma$ of size at most ℓ each, there are at most L messages $x \in \Sigma^k$ so that $C(x)_i \notin S_i$ for at most an α fraction of the coordinates $i \in [n]$. A special case of list recovery with lists S_i of size one is called *list decoding*: we refer to $(\alpha, 1, L)$-list recovery as (α, L)-list decoding.

List recoverable codes were first studied in the context of list decoding and soft decoding. The celebrated Guruswami-Sudan list decoding algorithm for Reed-Solomon codes [GS99] is in fact a list recovery algorithm, as are several more recent list decoding algorithms for variants of Reed-Solomon codes [GR08, GW13, Kop15, GX13]. Initially, list recoverable codes were used as stepping stones towards constructions of list decodable codes [GI01, GI02, GI03, GI04]. Since then, list recoverable codes have found additional applications in theoretical computer science in areas such as randomness extractors [Tre01, TZS06, TZ04, GUV09], group testing [INR10, GNP+13], compressed sensing [NPR12], and collision-resistant hashing [HIOS15].

*fbrett@cis.upenn.edu. Department of Computer Science, University of Pennsylvania
†noga@cs.haifa.ac.il. Department of Computer Science, University of Haifa
‡marykw@stanford.edu. Departments of Computer Science and Electrical Engineering, Stanford University

ISSN 1433-8092

It is well known that $1-\alpha$ is the *list recovery capacity*, in the sense that there exist codes of rate approaching $1-\alpha$ that are (α, ℓ, L)-list recoverable with small output list size L (independent of the codeword length n), and on the other hand, any code of rate larger than $1-\alpha$ must have output list size L that is exponential in n.

The focus of this paper is on *local* list recovery. Locality is another frequent desideratum in coding theory. Informally, a code "exhibits locality" if information about a single coordinate x_i of a message x of C can be determined locally from only a few coordinates of a corrupted version of $C(x)$. Locality, and in particular the notion of local list recovery that we will define below, has been implicit in theoretical computer science for decades. For example, local list decoding algorithms are at the heart of algorithms in cryptography [GL89], learning theory [KM93], average-to-worst-case reductions [Lip90], and hardness amplification [BFNW93, STV01].

The idea of local list recovery is as follows: given a message index $i \in [k]$, a local list recovery algorithm should produce a list of L possible symbols that could appear at that index. The catch is that we require that the lists returned are consistent across indices.

Formally, a local list recovery algorithm returns a list $A_1, \ldots, A_L$ of randomized local algorithms. Each of these local algorithms A_j takes a message index $i \in [k]$ as input and has oracle access to the input lists $S_1, \ldots, S_n$. The algorithm A_j then makes at most Q queries to this oracle (that is, it sees all elements in at most Q different input lists S_i), and it must return a guess for x_i, where x is a message whose encoding $C(x)$ agrees with many of the input lists. The guarantee is that for all such messages x—that is, for all x whose encoding $C(x)$ agrees with all but α-fraction of the input lists—there exists (with high probability) some A_j so that for all i, $A_j(i) = x_i$ with probability at least $2/3$. The parameter Q is called the *query complexity* of the local list recovery algorithm. On our way to constructing locally list recoverable codes, we will construct *approximately* locally list recoverable codes. We say that a code is approximately locally list recoverable if the local algorithms $A_1, \ldots, A_L$ described above may fail to correctly decode a small constant fraction of the message coordinates.

One reason to study local list recoverability is that list recovery is a useful building block in the construction of list decodable codes. In particular, the problem of constructing *high rate locally list recoverable codes* (of rate arbitrarily close to 1, and in particular non-decreasing as a function of ℓ) has been sought after for some time, because such codes would have implications in local and global list decoding.

In this work, we show that the tensor product of a high-rate globally list recoverable code is approximately locally list recoverable. Instantiating this with known constructions of high-rate globally list recoverable codes, and using a few transformations, we obtain the first *capacity-achieving* locally list decodable codes, as well as the first capacity-achieving globally list decodable codes with *nearly linear time* list decoding algorithm. Our techniques are inspired by the list decoding algorithm of [GGR11] for tensor codes, and our main observation is that this algorithm—with a few tweaks—can be made local.

1.1 Results

From global to approximately-local list recovery. Our main technical contribution is showing that the tensor product of a high-rate globally list recoverable code is (approximately) locally list recoverable. Given a *linear code* (i.e., a linear map) $C : \mathbb{F}^k \to \mathbb{F}^n$ over some finite field $\mathbb{F}$, consider the *tensor product code* $C \otimes C : \mathbb{F}^{k \times k} \to \mathbb{F}^{n \times n}$; we will define the tensor product formally in Definition 2.11, but for now, we will treat the codewords of $C \otimes C$ as $n \times n$ matrices with the

constraints that the rows and columns are all codewords of the base code C.

Informally, our main result shows that if C is globally list recoverable (and of arbitrarily high rate) then $C \otimes C$ is approximately locally list recoverable, in the sense described above, with roughly the same parameters, and with query complexity on the order of n (the codeword length of C). Note that the query complexity is about the square root of the codeword length of $C \otimes C$ which is n^2. The query complexity can be further reduced by applying the tensor product operation iteratively. Specifically, by taking the t-th tensor power, the resulting code $C^{\otimes t}$ of codeword length $N := n^t$ is approximately locally list recoverable with query complexity roughly $n = N^{1/t}$.

We state this main result below as Theorem 1.1, and prove it in Section 4. In the theorem statement, one should think of all parameters $\delta, \alpha, \varepsilon, L, t$, and consequently also s, as constants (or more generally, as slowly growing functions of n). In that case, Theorem 1.1 says that if C is (α, ℓ, L)-globally list recoverable, then the tensor product $C^{\otimes t}$ is ε-approximately $(\Omega(\alpha), \ell, L^{O(1)})$-locally list recoverable with query complexity $O(n) = O(N^{1/t})$.

Theorem 1.1 (From global to approximately-local)**.** *The following holds for any $\delta, \alpha, \varepsilon > 0$, $L \geq 1$, and $s = \mathrm{poly}(1/\delta, 1/\alpha, 1/\varepsilon, \log L)$. Suppose that $C : \mathbb{F}^k \to \mathbb{F}^n$ is a linear code of relative distance δ that is (α, ℓ, L)-globally list recoverable. Then $C^{\otimes t} : \mathbb{F}^{k^t} \to \mathbb{F}^{n^t}$ is ε-approximately $(\alpha \cdot s^{-t^2}, \ell, L^{s^{t^2}})$-locally list recoverable with query complexity $n \cdot s^{t^2}$.*

Basic list recovery transformations. To apply Theorem 1.1, in Section 3 we first present a few simple, yet powerful, list recovery transformations.

In more detail, in our first "approximately-local to local" transformation (Lemma 3.1) we observe that the "approximate" restriction can be eliminated by pre-encoding the message with a *locally decodable code* C' before encoding it with the approximately locally list recoverable code C. This way, instead of directly querying C (which may give the wrong answer a constant fraction of the time), we use the outer locally decodable code C' to query C: this still does not use too many queries, but now it is robust to a few errors. A similar transformation was noted in [BEAT] in the context of local list decoding.

The second "high-rate to capacity-achieving" transformation (Lemma 3.2) relies on an expander-based transformation of Alon, Edmunds, and Luby [AEL95, AL96]. This tranformation has been used before in this context; in particular, as previously observed by [GI03, KMRS17, GKO$^+$18], this technique can transform a high-rate locally list recoverable code into a capacity-achieving locally list recoverable code.

Finally, in our third "local to nearly-linear-time" transformation (Lemma 3.4) we observe that locally list recoverable codes straightforwardly extend to nearly-linear time globally list recoverable codes, simply by running the local algorithm on each coordinate.

Instantiations. To obtain our main results, we instantiate the transformations described above with known constructions of high-rate globally list recoverable codes. As the tensor operation inflates the output list size, we require our base code to have small (constant or very slowly growing) output list size. We also need the base code to be linear to get a handle on the rate of the tensor product.

In the first instantiation, Theorem 1.2 below, we just use a (non-efficient) random linear code. This gives capacity-achieving locally list recoverable codes with query complexity n^β for any constant $\beta > 0$, and with constant alphabet and output list sizes, although without an explicit construction or efficient list recovery algorithm.

Theorem 1.2. *For any constants $\rho \in [0,1]$, $\varepsilon, \beta > 0$, and $\ell \geq 1$ there exists an infinite family of codes $\{C_n\}_n$ such that the code C_n has block length n, alphabet size $O(1)$, rate ρ, and is $(1-\rho-\varepsilon, \ell, O(1))-$locally list recoverable with query complexity n^β.*

We prove Theorem 1.2 in Section 5.1. As a special case, this theorem gives the first capacity-achieving locally list decodable codes with $o(n)$ queries.

The second instantiation, Theorem 1.3 does yield efficient encoding and list recovery (in nearly-linear time) as well as locality. It uses a modification of the algebraic geometry subcodes studied in [GX13, GK16b] as the initial code. These latter codes have constant alphabet size, but slightly super-constant output list size (depending on $\log^* n$), which means that our construction will as well.

Theorem 1.3. *For any constants $\rho \in [0,1]$, $\varepsilon, \beta > 0$, and $\ell \geq 1$ there exists an infinite family of codes $\{C_n\}_n$, where C_n has block length n, alphabet size $O(1)$, rate ρ, and is $(1-\rho-\varepsilon, \ell, L)$-locally list recoverable with query complexity n^β for $L = \exp\exp\exp(\log^* n)$. Moreover, C_n is encodable and globally list recoverable in time $n^{1+O(\beta)}$.*

Theorem 1.3 is proven in Section 5.2. As a special case, the above theorem gives the first capacity-achieving globally list decodable codes with list decoding algorithm running in time $o(n^2)$.

Our final instantiation, Theorem 1.4, uses as the initial code the same algebraic geometry codes as the previous instantiation, but in a different regime of parameters. This gives *sub-polynomial* query complexity of $n^{o(1)}$, albeit with larger alphabet and output list sizes.

Theorem 1.4. *For any constants $\rho \in [0,1]$, $\varepsilon > 0$, and $\ell \geq 1$ there exists an infinite family of codes $\{C_n\}_n$, where C_n has block length n, alphabet size $n^{o(1)}$, rate ρ, and is $(1-\rho-\varepsilon, \ell, n^{o(1)})$-locally list recoverable with query complexity $n^{o(1)}$. Moreover, C_n is encodable and globally list recoverable in time $n^{1+o(1)}$.*

Theorem 1.4 is proven in Section 5.3, While we have not made an effort to optimize the $o(1)$ term in the exponent in the above theorem, we note that it is quite slowly decreasing (on the order of $1/\log\log n$).

Finally, we note that our approach is modular; given as an ingredient any high-rate (efficiently) globally list recoverable code, it yields a capacity-achieving locally (and nearly-linear time) list recoverable code with comparable parameters. Any improvements in these ingredient codes (for example, in the output list size of explicit linear high-rate globally list recoverable codes, which is nearly constant but not quite constant) would translate immediately into improvements in our constructions.

1.2 Related work

For those familiar with the area, it may be somewhat surprising that the results described above were not known before: indeed, we know of locally list recoverable codes, and we also know of capacity-achieving globally list recoverable codes. One might think that our result is lurking implicitly in those earlier works. However, as discussed below, it turns out that it is not so straightforward, and existing techniques for locally or globally list recoverable codes do not seem to work for this problem. Next we elaborate on these prior lines of work.

Local list recovery. Local list decoding first arose outside of coding theory, motivated by applications in complexity theory. For example, the Goldreich-Levin theorem in cryptography [GL89] and the Kushilevitz-Mansour algorithm in learning theory [KM93] can be interpreted as local list decoding algorithms for Hadamard codes. Later, Sudan, Trevisan and Vadhan [STV01], motivated by applications in pseudorandomness, gave an algorithm for locally list decoding Reed-Muller codes. Similar ideas were used later for list decoding *lifted codes* [GK16a] and *multiplicity codes* [Kop15], which can be viewed as high-rate variants of Reed-Muller codes.

As observed recently in [GKO+18], all the aforementioned local list decoding algorithms can be used for local list recovery as well. However, all these algorithms work only up to the *Johnson bound.* In the setting of list recovery, the Johnson bound implies that the rate of the code must be at most $1/\ell$. Thus this approach does not seem to give capacity-achieving locally list recoverable codes or even high-rate ones, and the Johnson bound appears to be a fundamental bottleneck for these techniques.

Global list recovery up to capacity. This line of work started with the celebrated work of Guruswami and Rudra [GR08], showing that folded Reed-Solomon codes achieve list decoding capacity. Since then, there has been a long line of work [Gur10, GW13, DL12, Kop15, GX12a, GX13, GK16b, GX14], aimed at reducing the alphabet and output list sizes, and improving the speed of the list decoding algorithm.

In many cases, the above list decoding algorithms extend also to list recovery. However, all these algorithms are very global: they are all based on finding some interpolating polynomial, and finding this polynomial requires querying almost all of the coordinates. Thus, it is not at all obvious how to tweak these sorts of algorithms to obtain locally list recoverable codes.

Finally, we mention the work of [HW18], which constructed capacity-achieving list recoverable codes based on expander graphs. While that construction is not explicitly local, it is not as clearly global as those previously mentioned (indeed, expander codes are known to have some locality properties [HOW15]). However, that work could only handle list recovery in the presence of erasures—that is, the setting in which the locations of the errors are known—and adapting it to handle errors seems like a challenging task.

Thus, even with a great deal of work on locally list recoverable codes, and on capacity-achieving globally list recoverable codes, it was somehow unclear how to follow those lines of work to obtain our results. Instead our work follows a different approach, based on the techniques of [GGR11] for list decoding tensor codes.

Finally, we note that the local *testing* properties of tensor codes have been extensively studied [BS06, Val05, CR05, DSW06, GM12, BV09, BV15, Vid15, Mei09, Vid13]. To the best of our knowledge, ours is the first work to explicitly study the local (list) *decodability* of tensor codes, rather than local testability. [1]

Subsequent work. In the follow-up work [KRSW18], involving a subset of the current authors, it was shown that *high-rate* multiplicity codes (one of the high-rate variants of Reed-Muller codes)

[1] We note that the work [MV05] implicitly studied the local list recovery properties of the low degree extension code (which can be viewed as a special instance of tensor codes) in the context of derandomization. The approach in [MV05] is similar to ours, except that they considered only the special case where all the lists on axis-parallel lines are the same, and there are no errors (which was appropriate for their application). Furthermore, that work was interested in locality but not efficiency, and so it did not give an efficient local list recovery algorithm.

are locally list recoverable. The main ingredient in obtaining this result was showing that high-rate univariate multiplicity codes are globally list recoverable with *constant* output list size. In contrast, prior work only obtained an output list size that is polynomial in the codeword length, which was a main barrier for bypassing the Johnson bound for local list recovery of multivariate multiplicity codes. In addition to this, several other modifications to the local list recovery algorithm of [STV01, Kop15] were required in order to make it work in the high-rate regime.

As a corollary, using the basic list recovery transformations described above, the above result gave an improvement over our main Theorems 1.3 and 1.4. In more detail, in the polynomial query complexity regime, the result of [KRSW18] reduced the output list size L in Theorem 1.3 to a constant. More significantly, in the sub-polynomial query complexity regime, it reduced the $n^{o(1)}$ term in Theorem 1.4 to $\widetilde{\exp}(\log^{3/4} n)$. Determining the exact query complexity of capacity-achieving locally list recoverable (or list decodable) codes remains an open problem.

1.3 Techniques

We end the introduction with a high-level overview of the proof of the "global to approximately-local" result, Theorem 1.1. As mentioned above, this algorithm is inspired by the analysis given by Gopalan, Guruswami, and Raghevendra in [GGR11] for the list decoding radius of tensor codes.

In more detail, in [GGR11] it is shown that the two-dimensional tensor code $C \otimes C$ is roughly as list decodable as C is. That work was primarily focused on combinatorial results about list decoding radius, but their analysis was algorithmic, and it is these algorithmic insights that we leverage here. Specifically, our main contribution is to observe that their analysis for two-dimensional tensor can be phrased in the language of approximate local list decoding, and moreover it extends also to higher-dimensional tensor products with lower query complexity. We further observe that the analysis straightforwardly extends to the setting of list recovery.

To understand the intuition, let us describe the algorithm just for $C \otimes C$, although our final results will require a higher tensor power $C^{\otimes t}$. For simplicity, additionally assume that there are no errors, *i.e.*, $\alpha = 0$. Recall that the tensor product of a linear code $C : \mathbb{F}^k \to \mathbb{F}^n$ is the code $C \otimes C : \mathbb{F}^{k \times k} \to \mathbb{F}^{n \times n}$ whose codewords are all $n \times n$ matrices with the constraints that the rows and columns are all codewords of the base code C.

Following [GGR11], the local list recovery algorithm A for $C \otimes C$ first chooses m random columns of $[n] \times [n]$ for a small integer m. These each correspond to codewords in C, and A runs C's global list recovery algorithm on them to obtain output lists $\mathcal{L}_1, \ldots, \mathcal{L}_m$, of size at most L each, on each of these columns. Finally, for any choice of one codeword per column $c_1 \in \mathcal{L}_1, c_2 \in \mathcal{L}_2, \ldots, c_m \in \mathcal{L}_m$, it outputs a local algorithm $A_{(c_1,\ldots,c_m)}$ indexed by $(c_1, \ldots, c_m)$. Notice that the query complexity is mn, which is roughly the square root of the codeword length of $C \otimes C$ (which is n^2), while the output list size (the number of local algorithms) is L^m. We view $c_1, \ldots, c_m$ as "advice" that fixes the value of the codeword on the chosen m columns.

Next we describe the local algorithm $A_{(c_1,\ldots,c_m)}$ indexed by $(c_1, \ldots, c_m)$, on input $(i, j) \in [n] \times [n]$; this algorithm is illustrated in Figure 1.[2] Recall that $A_{(c_1,\ldots,c_m)}$ is allowed to query the input lists at every coordinate, and must produce a guess for the codeword value indexed by (i, j). Let $v_1, \ldots, v_m$ denote the values of the codewords $c_1, \ldots, c_m$ on the i'th position. The algorithm $A_{(c_1,\ldots,c_m)}$ runs C's global list recovery algorithm once more on the i'th row to obtain another output list $\mathcal{L}$. Now

[2]The algorithm we describe decodes codeword symbols instead of messages symbols, but since the tensor code can be made systematic (*i.e.*, the message is part of the codeword) this algorithm can also decode message symbols.

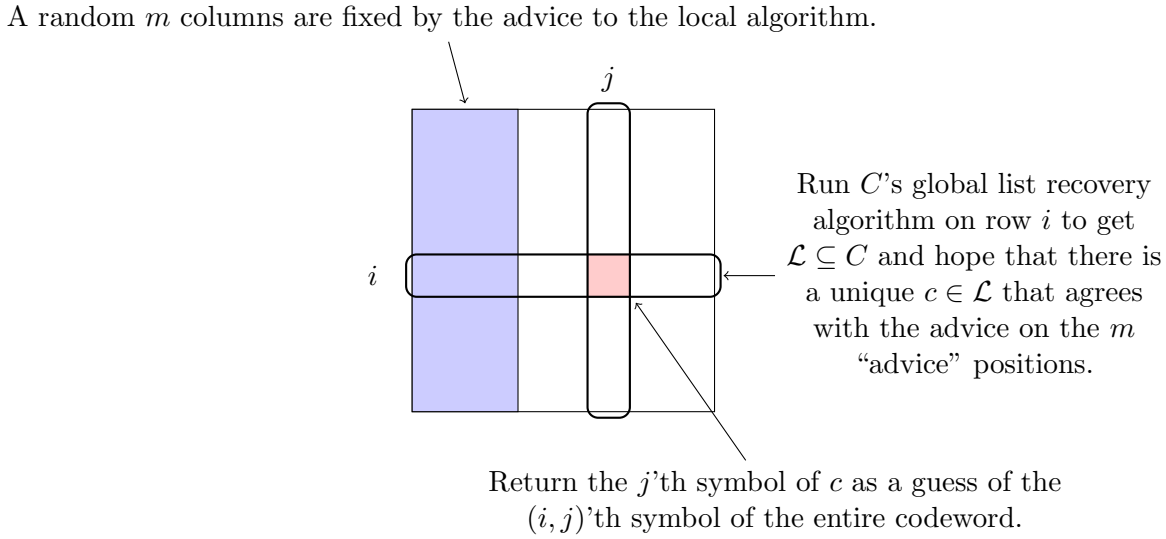

Figure 1: The local algorithm $A_{(c_1,\ldots,c_m)}$, for $C \otimes C$ on input $(i, j) \in [n] \times [n]$. Here, $c_0, \ldots, c_m$ are "advice:" $c_i \in \mathcal{L}_i$ is one of L possible codewords for the i'th randomly selected column. The set of selected columns is random, but we have show them as the first m columns for simplicity.

the algorithm chooses an arbitrary codeword $c \in \mathcal{L}$ that agrees with the advice $v_1, \ldots, v_m$ on the chosen columns, if such a c exists. Finally, the j'th symbol of c is $A_{(c_1,\ldots c_m)}$'s guess for the (i, j) symbol of the tensor codeword.

Now, observe that the guess for (i, j) would be correct if there is at most one codeword in $\mathcal{L}$ that agrees with the advice $v_1, \ldots, v_m$ on the chosen columns, so no confusion arises. Assuming that the code C has relative distance at least δ, any pair of codewords in $\mathcal{L}$ differ in at least δ fraction of the coordinates. Recalling that the m columns were chosen completely at random, a Chernoff bound, followed by a union bound over all elements in $\mathcal{L}$, imply that the value on the m columns will disambiguate any pair of codewords in $\mathcal{L}$ with high probability, assuming that $m \approx (\log L)/\delta^2$.

The above idea gives a code of length N that is locally list recoverable with query complexity on the order of $\sqrt{N}$. This algorithm for $C \otimes C$ extends to $C^{\otimes t}$, with query complexity roughly $N^{1/t}$. The trade-off is that the output list size also grows with t. Thus, as we continue to take tensor powers, the locality improves, while the output list size degrades; this allows for the trade-off between locality and output list size that we obtain in Theorems 1.3 and 1.4. The algorithm can be made to work also in the presence of errors, except that it may fail on small constant fraction of coordinates (e.g., when a whole row is corrupted). Consequently, we only obtain an approximately local list recovery algorithm that decodes correctly most, but not all, of the coordinates.

Organization. The rest of the paper is organized as follows. In Section 2 we set up notation and definitions. In Section 3 we present some basic list recovery transformations (from approximately-local to local, from high-rate to capacity-achieving, and from local to nearly-linear-time). In Section 4 we present our main transformation from global to approximately-local list recovery which proves our Theorem 1.1. Finally, in Section 5 we instantiate our transformations with known constructions of high-rate globally list recoverable codes, thus proving our main Theorems 1.2, 1.3, and 1.4.

2 Preliminaries

For a prime power q we denote by $\mathbb{F}_q$ the finite field of q elements. For any finite alphabet Σ and for any pair of strings $x, y \in \Sigma^n$, the relative distance between x and y is the fraction of coordinates $i \in [n]$ on which x and y differ, and is denoted by $\mathrm{dist}(x, y) := |\{i \in [n] : x_i \neq y_i\}| / n$. For a positive integer ℓ we denote by $\binom{\Sigma}{\ell}$ the collection of all subsets of Σ of size ℓ, and for any string $x \in \Sigma^n$ and tuple $S \in \binom{\Sigma}{\ell}^n$ we denote by $\mathrm{dist}(x, S)$ the fraction of coordinates $i \in [n]$ for which $x_i \notin S_i$, that is, $\mathrm{dist}(x, S) := |\{i \in [n] : x_i \notin S_i\}| / n$. If $\mathrm{dist}(x, S) \leq \alpha$, we say that x is α-close to S. For a string $x \in \Sigma^n$ and a set $T \subseteq [n]$, we use $x|_T \in \Sigma^{|T|}$ to denote the restriction of x to the coordinates in T. Throughout the paper, we use $\exp(n)$ to denote $2^{\Theta(n)}$, and whenever we use log, it is base 2, unless noted otherwise.

2.1 Error-correcting codes

Let Σ be a finite alphabet, and k, n be positive integers (the message length and the block length, respectively). An (error-correcting) code is an injective map $C : \Sigma^k \to \Sigma^n$. The elements in the domain of C are called messages, and the elements in the image of C are called codewords. We say that C is systematic if the message is a prefix of the corresponding codeword, i.e., for every $x \in \Sigma^k$ there exists $z \in \Sigma^{n-k}$ such that $C(x) = (x, z)$.

If $\mathbb{F}$ is a finite field, and Σ is a vector space over $\mathbb{F}$, we say that C is $\mathbb{F}$-linear if it is a linear transformation over $\mathbb{F}$ between the $\mathbb{F}$-vector spaces Σ^k and Σ^n. If $\Sigma = \mathbb{F}$ and C is $\mathbb{F}$-linear, we simply say that C is linear. The generating matrix of a linear code $C : \mathbb{F}^k \to \mathbb{F}^n$ is a matrix $G \in \mathbb{F}^{n \times k}$ such that $C(x) = G \cdot x$ for any $x \in \mathbb{F}^k$. Note that a linear code can be made systematic by applying a Gaussian elimination on the generating matrix G.

The rate of a code $C : \Sigma^k \to \Sigma^n$ is the ratio $\rho := \frac{k}{n}$. The relative distance $\mathrm{dist}(C)$ of C is the minimum $\delta > 0$ such that for every pair of distinct messages $x, y \in \Sigma^k$ it holds that $\mathrm{dist}(C(x), C(y)) \geq \delta$. If $C : \Sigma^k \to \Sigma^n$ has relative distance δ then for any parameter $\alpha < \delta/2$, and for any received word $w \in \Sigma^n$, there is at most one message $x \in \Sigma^k$ which satisfies $\mathrm{dist}(C(x), w) \leq \alpha$.

2.2 List recoverable codes

List recovery is a generalization of the standard error-correction setting where each entry w_i of the received word w is replaced with a list S_i of ℓ possible symbols of Σ. Formally, for a parameter $\alpha \in [0, 1]$ and integers ℓ, L we say that a code $C : \Sigma^k \to \Sigma^n$ is (α, ℓ, L)-list recoverable if for any tuple $S \in \binom{\Sigma}{\ell}^n$ there are at most L different messages $x \in \Sigma^k$ so that $\mathrm{dist}(C(x), S) \leq \alpha$. We say that C is (α, L)-list decodable if it is $(\alpha, 1, L)$-list recoverable.

For $\alpha \in [0, 1]$ let

$$H(\alpha) = \alpha \log(1/\alpha) + (1 - \alpha) \log(1/(1 - \alpha))$$

denote the binary entropy function.

We will use the following theorem about the list recoverability of random linear codes.

Theorem 2.1 ([Gur01], Lemma 9.6)**.** *For any prime power q, integers $1 \leq \ell \leq q$ and $L > \ell$, parameters $0 \leq \alpha \leq 1$ and*

$$\rho < \frac{1}{\log q} \cdot \left[(1 - \alpha) \cdot \log(q/\ell) - H(\alpha) - H(\ell/q) \cdot \frac{q}{\log_q(L + 1)} \right],$$

and sufficiently large n, a random linear code $C : \mathbb{F}_q^k \to \mathbb{F}_q^n$ of rate ρ is (α, ℓ, L)-list recoverable with probability at least $1 - \exp(-n)$.

Corollary 2.2. *For any $\rho \in [0,1]$, $\varepsilon > 0$, $\ell \geq 1$, and for sufficiently large prime power q and integer n, a random linear code $C : \mathbb{F}_q^k \to \mathbb{F}_q^n$ of rate ρ has relative distance at least $1 - \rho - \varepsilon$, and is $(1 - \rho - \varepsilon, \ell, q^{O(\ell/\varepsilon)})$-list recoverable, with probability at least $1 - \exp(-n)$.*

Proof. The relative distance follows by the Gilbert-Varshamov bound [Gil52, Var57]. For the list recovery properties, apply Theorem 2.1 with $\alpha = 1 - \rho - \varepsilon$,

$$q \geq \max\{(1-\rho-\varepsilon)^{-c_0(1-\rho-\varepsilon)/\varepsilon}, (\rho+\varepsilon)^{-c_0(\rho+\varepsilon)/\varepsilon}, \ell^{c_0/\varepsilon}\},$$

and $L = q^{c_0\ell/\varepsilon}$ for some absolute constant c_0, and note that in this setting of parameters,

$$\begin{aligned} & \frac{1}{\log q} \cdot \left[(\rho+\varepsilon) \cdot \log(q/\ell) - H(1-\rho-\varepsilon) - H(\ell/q) \cdot \frac{q}{c_0\ell/\varepsilon} \right] \\ \geq \;& \rho + \varepsilon - \frac{\log \ell}{\log q} - \frac{(1-\rho-\varepsilon)\log(1/(1-\rho-\varepsilon))}{\log q} - \frac{(\rho+\varepsilon)\log(1/(\rho+\varepsilon))}{\log q} - O(\varepsilon/c_0) \\ \geq \;& \rho + \varepsilon - O(\varepsilon/c_0). \end{aligned}$$

Thus, the corollary holds for a sufficiently large constant c_0. □

2.3 Locally decodable codes

Intuitively, a code C is said to be locally decodable if, given a codeword $C(x)$ that has been corrupted by some errors, it is possible to decode any coordinate of the corresponding message x by reading only a small part of the corrupted version of $C(x)$. Formally, it is defined as follows.

Definition 2.3 (Locally decodable code)**.** A code $C : \Sigma^k \to \Sigma^n$ is α-locally decodable with query complexity Q if there exists a randomized algorithm A that satisfies the following requirements:

- **Input:** A takes as input a coordinate $i \in [k]$, and also gets oracle access to a string $w \in \Sigma^n$ that is α-close to some codeword $C(x)$.
- **Query complexity:** A makes at most Q queries to the oracle w.
- **Output:** A outputs x_i with probability at least $\frac{2}{3}$.

Remark 2.4. By definition it holds that $\alpha < \mathrm{dist}(C)/2$. The above success probability of $\frac{2}{3}$ can be amplified using sequential repetition, at the cost of increasing the query complexity and running time. Specifically, amplifying the success probability to $1 - \exp(-t)$ requires increasing the query complexity and running time by a multiplicative factor of $O(t)$.

The following theorem shows the existence of high-rate locally decodable codes with sub-polynomial query complexity of the form $n^{o(1)}$.

Theorem 2.5 ([KMRS17], Theorem 1.3)**.** *For any constant $\rho \in [0,1]$ and $\varepsilon > 0$ there exist a constant s and an infinite family of codes $\{C_n\}_n$ that satisfy the following.*

- *C_n is an $\mathbb{F}_2$-linear code of block length n and alphabet size 2^s.*

- *C_n has rate ρ and relative distance at least $1-\rho-\varepsilon$.*
- *C_n is $\frac{1-\rho-\varepsilon}{2}$-locally decodable with query complexity $n^{o(1)}$ in time $n^{o(1)}$.*
- *C_n is encodable in time $n^{1+o(1)}$.*

Remark 2.6. The encoding time is not stated explicitly in [KMRS17]. However, the construction in [KMRS17] proceeds by first considering multiplicity codes of high-rate $1-o(1)$, query complexity $n^{o(1)}$, and sub-constant decoding radius $n^{-o(1)}$ ([KMRS17, Lemma 3.3]), and then amplifying the decoding radius to a constant using the AEL transformation ([KMRS17, Lemma 3.2]). The encoding time of $n^{1+o(1)}$ for multiplicity codes in this regime follows from [Cox18], using the choice of parameters made in [KMRS17, Lemma 3.3], while the encoding time of $n^{1+o(1)}$ for the AEL transformation can be deduced from the proof of [KMRS17, Lemma 3.2] (see also Lemma 3.2 in the current paper that shows a similar transformation for the local list recovery setting).

Theorem 1.3 in [KMRS17] applies to *locally correctable codes* instead of locally decodable codes. The difference is that for the former the local correction algorithm is required to decode codeword coordinates as opposed to message coordinates. However, since the encoding map described above is systematic, the local correction algorithm decodes also message coordinates (see discussion in [KMRS17, Section 1.3]).

2.4 Locally list recoverable codes

The following definition generalizes the notion of locally decodable codes to the setting of list recovery. In this setting the local list recovery algorithm is required to output in an implicit sense all messages whose corresponding codewords are consistent with most of the input lists.

Definition 2.7 (Locally list recoverable code)**.** A code $C : \Sigma^k \to \Sigma^n$ is (α, ℓ, L)-locally list recoverable with query complexity Q if there exists a randomized algorithm A that satisfies the following requirements:

- **Preprocessing:** On input 1^n, A outputs L randomized algorithms $A_1, \ldots, A_L$.
- **Input:** Each A_j takes as input a coordinate $i \in [k]$, and also gets oracle access to a tuple $S \in \binom{\Sigma}{\ell}^n$.
- **Query complexity:** Each A_j makes at most Q queries to the oracle S.
- **Output:** For every codeword $C(x)$ that is α-close to S, with probability at least $\frac{2}{3}$ over the randomness of A the following event happens: there exists some $j \in [L]$ such that for all $i \in [k]$,
$$\Pr\left[A_j(i) = x_i\right] \geq \frac{2}{3}, \tag{1}$$
where the probability is over the internal randomness of A_j.

We say that A has preprocessing time T_{pre} if A outputs the description of the algorithms $A_1, \ldots, A_L$ in time at most T_{pre}, and has running time T if each A_j has running time at most T. As before, we say that the code C is (α, L)-locally list decodable with query complexity Q if it is $(\alpha, 1, L)$-locally list recoverable with query complexity Q.

Remark 2.8. The success probability of A can be amplified by sequentially repeating A and outputting the union of all output lists, at the cost of increasing the output list size and preprocessing time. Specifically, amplifying the success probability to $1 - \exp(-t)$ requires increasing the output list size and preprocessing time by a multiplicative factor of $O(t)$.

Remark 2.9. In our definition of a locally list recoverable code, we are concerned with recovering message symbols x_i for $i \in [k]$. Another definition might be to recover any *codeword* symbol $C(x)_i$ for $i \in [n]$. Since all of the codes we consider can be made systematic, this second definition would be stronger than Definition 2.7. Most of our techniques work for this stronger definition as well; the only one that does not work for this stronger definition is the reduction from approximately-local to local (Lemma 3.1).

Finally, we define an approximate version of local list recovery. In the following definition, the local algorithms A_j are deterministic, but each of them has to work on only a $1 - \varepsilon$ fraction of the coordinates $i \in [k]$.

Definition 2.10 (Approximately locally list recoverable code)**.** A code $C : \Sigma^k \to \Sigma^n$ is ε-**approximately** (α, ℓ, L)-locally list recoverable with query complexity Q if there exists a randomized algorithm A that satisfies the following requirements:

- **Preprocessing:** On input 1^n, A outputs L *deterministic* algorithms $A_1, \ldots, A_L$.
- **Input:** Each A_j takes as input a coordinate $i \in [k]$, and also gets oracle access to a tuple $S \in \binom{\Sigma}{\ell}^n$.
- **Query complexity:** Each A_j makes at most Q queries to the oracle S.
- **Output:** For every codeword $C(x)$ that is α-close to S, with probability at least $\frac{2}{3}$ over the randomness of A the following event happens: there exists some $j \in [L]$ such that for all $i \in [k]$,
$$\Pr_{i \in [k]} \left[A_j(i) = x_i \right] \geq 1 - \varepsilon. \tag{2}$$

As in Definition 2.7, we say that A has preprocessing time T_{pre} if A outputs the description of the algorithms $A_1, \ldots, A_L$ in time at most T_{pre}, and has running time T if each A_j has running time at most T. We note that Remark 2.8 holds also for approximately locally list recoverable codes, so we may amplify the success probability of $2/3$ to $\exp(-t)$ at the cost of multiplying the list size and pre-processing time by a factor of $O(t)$.

2.5 Tensor codes

Our construction is based on the tensor product operation, defined as follows.

Definition 2.11 (Tensor codes)**.** Let $C_1 : \mathbb{F}^{k_1} \to \mathbb{F}^{n_1}$, $C_2 : \mathbb{F}^{k_2} \to \mathbb{F}^{n_2}$ be linear codes, and let $G_1 \in \mathbb{F}^{n_1 \times k_1}, G_2 \in \mathbb{F}^{n_2 \times k_2}$ be the generating matrices of C_1, C_2 respectively. Then the tensor code $C_1 \otimes C_2 : \mathbb{F}^{k_1 \times k_2} \to \mathbb{F}^{n_1 \times n_2}$ is given by $(C_1 \otimes C_2)(M) = G_1 \cdot M \cdot G_2^T$ for any $k_1 \times k_2$ matrix M over $\mathbb{F}$.

Note that the codewords of $C_1 \otimes C_2$ are $n_1 \times n_2$ matrices over $\mathbb{F}$ whose columns belong to the code C_1 and whose rows belong to the code C_2.

Fact 2.12. *Suppose that $C_1 : \mathbb{F}^{k_1} \to \mathbb{F}^{n_1}$, $C_2 : \mathbb{F}^{k_2} \to \mathbb{F}^{n_2}$ are linear codes of rates ρ_1, ρ_2 and relative distances δ_1, δ_2, respectively. Then the tensor code $C_1 \otimes C_2$ has rate $\rho_1 \cdot \rho_2$ and relative distance $\delta_1 \cdot \delta_2$. Moreover, if C_1, C_2 are encodable in times T_1, T_2, respectively, then $C_1 \otimes C_2$ is encodable in time $n_1 T_2 + n_2 T_1$.*

Proof. The rate and distance properties are well known (see e.g. [Sud01, DSW06]). The encoding time follows since one can encode a message $M \in \mathbb{F}^{k_1 \times k_2}$ by first encoding all rows of M using C_2, and then encoding all columns of the resulting matrix using C_1. □

For a linear code C, let $C^{\otimes 1} := C$ and $C^{\otimes t} := C \otimes C^{\otimes (t-1)}$. By induction on t we have the following.

Corollary 2.13. *Suppose that $C : \mathbb{F}^k \to \mathbb{F}^n$ is a linear code of rate ρ and relative distance δ. Then the tensor code $C^{\otimes t} : \mathbb{F}^{k^t} \to \mathbb{F}^{n^t}$ has rate ρ^t and relative distance δ^t. Moreover, if C is encodable in time T, then $C^{\otimes t}$ is encodable in time $t \cdot n^t \cdot T$.*

3 Basic list recovery transformations

We begin by presenting a some basic list recovery reductions showing how to transform an approximately locally list recoverable code into a locally list recoverable code (Section 3.1), then a locally list recoverable code of high-rate into a capacity-achieving locally list recoverable code (Section 3.2), and finally a locally list recoverable code into a nearly-linear-time globally list recoverable code (Section 3.3).

3.1 From approximately-local to local

We start with the transformation from approximately local list recovery to local list recovery. The following lemma shows that one can pre-encode a Q-query ε-approximately locally list recoverable code with a Q'-query ε-locally decodable code to obtain a $(Q \cdot Q')$-query locally list recoverable code with the same parameters.

Lemma 3.1 (From approximately-local to local)**.** *Suppose that $C : \Sigma^k \to \Sigma^n$ is ε-approximately (α, ℓ, L)-locally list recoverable with query complexity Q, and $C' : \Sigma^{k'} \to \Sigma^k$ is ε-locally decodable with query complexity Q'. Then the composition $C \circ C' : \Sigma^{k'} \to \Sigma^n$ given by $C \circ C'(x) = C(C'(x))$ is (α, ℓ, L)-locally list recoverable with query complexity $Q \cdot Q'$.*

Moreover, if the approximately local list recovery algorithm for C has preprocessing time T_{pre} and running time T, and the local decoding algorithm for C' has running time T', then the local list recovery algorithm for $C \circ C'$ has preprocessing time T_{pre} and running time $T \cdot T'$.

Proof. We recover a $(1 - \varepsilon)$-fraction of the coordinates using the approximately local list recovery algorithm for C, and correct the rest of the coordinates using the local decoding algorithm for C'.

In more detail, the local list recovery algorithm for $C \circ C'$ replaces each of the local algorithms A_j generated by the approximately local list recovery algorithm for C with a local algorithm A'_j, operating as follows: on input $i \in [k']$, the local algorithm A'_j first invokes the local decoding algorithm A' for C' on input i, and then invokes A_j on each of the queries of A' in $[k]$. (See Figure 2).

Correctness follows since A_j outputs the correct value on at least a $(1 - \varepsilon)$-fraction of the coordinates in $[k]$, and A' outputs the correct value with probability at least $2/3$ provided that at most

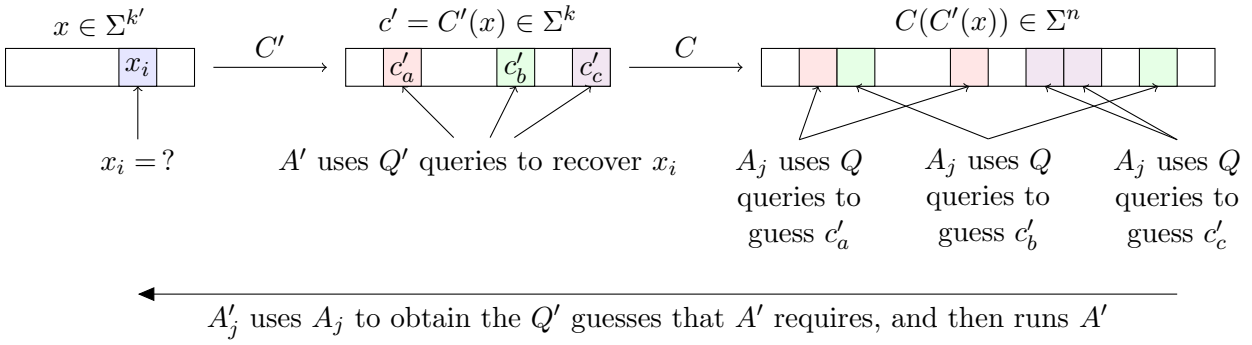

Figure 2: The local algorithm A'_j used in the proof of Lemma 3.1.

an ε-fraction of the codeword coordinates are corrupted. The query complexity is $Q \cdot Q'$, because for each of the Q' queries that A' would make, A'_j runs a copy of A_j which makes Q queries, and the running time is $T \cdot T'$ for the same reason. The preprocessing time is simply the preprocessing time for C's local list recovery algorithm. □

3.2 From high-rate to capacity-achieving

Next we turn to the transformation from a locally list recoverable code of high-rate into a capacity-achieving locally list recoverable code. To this end we use the Alon-Edmonds-Luby (AEL) distance amplification method [AEL95, AL96]. This technique was originally used in [AEL95, AL96] to improve error correction capabilities in the global unique decoding setting, and was later applied also in the list decoding/recovery regime [GI02, GR08, HW18]. More recent works have shown its usefulness for local decoding [KMRS17] local list recovery [GKO+18].

We shall use the following lemma from [GKO+18] which roughly says that given an "outer" code C of rate approaching 1 that is locally list recoverable from a tiny fraction of errors, and a small "inner" code C' that is a capacity-achieving globally list recoverable code, they can be combined using the AEL transformation to get a new code C_{AEL} with the following properties. On the one hand, C_{AEL} inherits the tradeoff between rate and error correction that C' enjoys; on the other hand, C_{AEL} does not use many more queries than C. Thus this procedure amplifies the distance from which we can list recover without significantly worsening other parameters. For completeness, we provide below a high-level description of the construction, and the reader is referred to [GKO+18, Section 7] for the full proof.

Lemma 3.2 (From high-rate to capacity-achieving)**.** *[GKO+18, Lemma 5.4.] There exists an absolute constant b_0 such that the following holds for any $\delta, \alpha, \varepsilon > 0$ and $t \geq (\delta \cdot \alpha \cdot \varepsilon)^{-b_0}$.*

Suppose that $C : (\Sigma^s)^k \to (\Sigma^s)^n$ is an outer code of rate $1 - \varepsilon$ and relative distance δ that is (α, ℓ, L)-locally list recoverable with query complexity Q, and $C' : \Sigma^s \to \Sigma^t$ is an inner code of rate ρ and relative distance $1-\rho-\varepsilon$ that is $(1-\rho-\varepsilon, \ell', \ell)$-globally list recoverable. Then there exists a code $C_{AEL} : (\Sigma^s)^k \to (\Sigma^t)^n$ of rate $\rho - \varepsilon$ and relative distance $1 - \rho - 2\varepsilon$ that is $(1 - \rho - 2\varepsilon, \ell', L)$-locally list recoverable with query complexity $Q \cdot \mathrm{poly}(t)$.

Suppose furthermore that C has encoding time T_{enc}, and the local list recovery algorithm for C has preprocessing time T_{pre} and running time T, and C' has encoding time T'_{enc} and list recovery time T'. Then C_{AEL} has encoding time $T_{enc} + n \cdot (T'_{enc} + \mathrm{poly}(t, \log n))$, and the local list recovery algorithm for C_{AEL} has preprocessing time T_{pre} and running time $T + Q \cdot (T' + \mathrm{poly}(t, \log n))$. Moreover, if C and C' are $\mathbb{F}$-linear then so is C_{AEL}.

Remark 3.3. The definition of a locally list recoverable code in [GKO+18, Definition 2.10] is slightly different from ours since in [GKO+18] the local list recovery algorithm is required to decode codeword coordinates as opposed to message coordinates, and must satisfy an additional soundness property which guarantees that with high probability, all local algorithms $A_1, \ldots, A_L$ compute an actual codeword. However, the proof of the above lemma goes through without change also for our definition of a locally list recoverable code.

Proof overview. (Included for completeness; see [GKO+18, Section 7] for a full proof). Given a message $x \in (\Sigma^s)^k$, we encode it first with the outer code $C : (\Sigma^s)^k \to (\Sigma^s)^n$, and then encode each symbol of the resulting codeword $c \in (\Sigma^s)^n$ using the inner code $C' : \Sigma^s \to \Sigma^t$. Clearly, the final rate would be the product of the rates of C and C' which is at least $\rho - \varepsilon$, we would like to show that final error correction radius would also be close to that of C', specifically $1 - \rho - 2\varepsilon$.

Suppose first that we have $1 - \rho - 2\varepsilon$ fraction of errors which are *randomly* chosen. Then by a Chernoff bound, we can say that almost all (except for at most an α fraction) the inner encodings will have at most a $1 - \rho - \varepsilon$ fraction of errors. Thus, we can list recover most of the inner encodings (except for at most an α fraction). Finally, we can list recover the outer code C from an α fraction of errors. Since we are interested in local list recovery, we only need to list recover those inner encodings which are queried by the local list recovery algorithm for C. Thus we will not lose much in locality, as long as the length of the inner encodings is small.

However, we need to deal with *adversarial* errors, not random errors, and so the analysis above might seem useless. Indeed, adversarial errors can easily completely wipe more than α fraction of inner encodings. To overcome this problem, we use samplers. Roughly speaking, a sampler is an n vertex bipartite d-regular graph in which the density of any subset T of right vertices is approximated by the value of $E(v, T)/d$ for a uniformly random left vertex v. We will use a fully explicit sampler that enables one to compute the list of neighbors of some given vertex in time $\mathrm{poly}(d, \log n)$.

We will choose the degree d of the regular bipartite graph (sampler) to be equal to the length t of the inner code. We associate each inner encoding with a left vertex of the graph and distribute its symbols to each of the neighbors on the right. The right vertices collect these symbols from their neighbors and repackage them as single symbol over a larger alphabet. This will be the final codeword which will have the same length n as C but over a slightly larger alphabet of Σ^t. The property of the sampler will ensure that whenever a $1 - \rho - 2\varepsilon$ fraction of symbols in the final codeword are corrupted, then after undoing the permutation of the sampler, almost all (except for at most a tiny α fraction) the inner encodings will have at most a $1 - \rho - \varepsilon$ fraction of errors. Now we can proceed as we did in the analysis of random errors.

3.3 From local to nearly-linear-time

Our final basic transformation is from local list recovery to nearly-linear-time global list recovery. We say that a code $C : \Sigma^k \to \Sigma^n$ is (α, ℓ, L)-globally list recoverable probabilistically if there exists

a probabilistic algorithm that on input $S \in \binom{\Sigma}{\ell}^n$, outputs a list of at most L messages, such that with probability at least $2/3$, the output list contains all x so that $\mathrm{dist}(C(x), S) \leq \alpha$.

Lemma 3.4 (From local to nearly-linear-time)**.** *Suppose that $C : \Sigma^k \to \Sigma^n$ is (α, ℓ, L)-locally list recoverable with query complexity Q, preprocessing time T_{pre}, and running time T. Then C is $(\alpha, \ell, O(L \log L))$-globally list recoverable probabilistically in time*

$$O\big(T_{pre} \cdot \log L + T \cdot (k \log k) \cdot (L \log^2 L)\big).$$

Proof. Suppose that C is as in the statement of the lemma. The proof outline is as follows: for each local algorithm A_j in the output list, we run A_j on each position $i \in [k]$ to obtain a message. This gives us a list of possible messages. There are a constant number of local algorithms A_j, and each call to a fixed A_j runs in sub-linear time. Since we call each A_j k times, the total running time is nearly linear in k. The only subtlety is that, since the local decoding procedure is probabilistic, we will need to amplify the probability of success to the point that we can take a union bound over the probability of failure in each coordinate and for each message in the output list.

In more detail, first note that by Remark 2.8, we may assume that the local list recovery algorithm A for C has failure probability at most $1/(101L)$, at the cost of increasing the output list size to $O(L \log L)$ and the preprocessing time to $O(T_{\text{pre}} \cdot \log L)$. By Remark 2.4, we may further assume that each local algorithm A_j in the output list of A fails with probability at most $1/(100kL)$, at the cost of increasing the running time of each A_j to $O(T \log(kL))$.

To globally list recover C we first run the local list recovery algorithm A for C. Then for each of the local algorithms A_j in the output list of A, we output a message that results by applying A_j on each of the k message coordinates. It can be verified that running time and output list size of this algorithm are as claimed.

Let $S \subseteq C$ be the list of codewords that are α-close to the input lists. For each $c \in S$, A returns a local algorithm A_j which corresponds to S with probability at least $1/(101 \cdot L)$, so we conclude that

$$|S| \leq L \cdot \left(1 - \frac{1}{101L}\right)^{-1} \leq 1.01 \cdot L.$$

By a union bound, the probability that there is some $c \in S$ so that no A_j corresponds to c is at most

$$|S| \cdot \frac{1}{101L} \leq \frac{1.01L}{101L} = \frac{1}{100}.$$

Next, conditioning on the event that all $c \in S$ have some corresponding local algorithm A_j, we apply a union bound over all L of these local algorithms, and all k coordinates that each of them might decode. We conclude that the probability that the global list recovery algorithm described above fails is at most

$$L \cdot k \cdot \frac{1}{100Lk} \leq \frac{1}{100}.$$

Thus, by a union bound over the two bad events, the probability that the global list recovery algorithm fails is at most $2/100 < 1/3$.

□

4 From global to approximately-local list recovery

We now turn to our main transformation from global list recovery to approximately local list recovery which proves our main Theorem 1.1. For the purposes of this section, it will be more natural to require that the approximately local list recovery algorithm recovers *codeword coordinates* as opposed to message coordinates. Formally, we assume that each local algorithm A_j receives as input a coordinate $i \in [n]$, and the requirement (2) is now replaced with the condition that

$$\Pr_{i \in [n]} \left[A_j(i) = (C(x))_i \right] \geq 1 - \varepsilon, \tag{3}$$

where the probability is over the choice of uniform random $i \in [n]$. Note that for a systematic code or rate ρ, Condition (3) with approximability parameter ε implies Condition (2) with approximability parameter ε/ρ. As all the codes that we will apply Theorem 1.1 to are high rate (in particular of rate greater than $1/2$), and can be made systematic, the difference between these definitions is negligible.[3]

Our main result in this section is the following lemma which implies Theorem 1.1.

Lemma 4.1 (From global to approximately-local)**.** *There exists an absolute constant b_0 such that the following holds for any $\delta, \alpha, \varepsilon > 0$ and $s \geq (\delta \cdot \alpha \cdot \varepsilon)^{-b_0}$, provided that ε is sufficiently small.*

Suppose that $C : \mathbb{F}^k \to \mathbb{F}^n$ is a linear code of relative distance δ that is (α, ℓ, L)-globally list recoverable. Then $C^{\otimes t} : \mathbb{F}^{k^t} \to \mathbb{F}^{n^t}$ is ε-approximately $(\alpha \cdot s^{-t^2}, \ell, L^{s^{t^2} \cdot \log^t L})$-locally list recoverable with query complexity $n \cdot (s^{t^2} \log^t L)$.

Moreover, if C can be list recovered in time $\mathrm{poly}(n)$, then the approximately local list recovery algorithm for $C^{\otimes t}$ has preprocessing time $O\left(\log(n) \cdot s^{t^2} \cdot \log^t L + L^{s^{t^2} \cdot \log^t L}\right)$ and running time $\mathrm{poly}(n) \cdot (s^{t^2} \log^t L)$.

Lemma 4.1 follows from Lemma 4.2 below which we view as our main technical lemma. Lemma 4.2 shows that the tensor product of an approximately locally list recoverable code with a globally list recoverable code results in an approximately locally list recoverable code with roughly the same performance. Lemma 4.1 then follows by applying the main technical lemma iteratively.

Lemma 4.2 (Main Technical Lemma)**.** *There exist absolute constants b_0 and d_0 such that the following holds for any $\delta', \alpha', \varepsilon > 0$ and $s \geq (\delta' \cdot \alpha' \cdot \varepsilon)^{-b_0}$, so that ε is sufficiently small.*

Suppose that $C : \mathbb{F}^k \to \mathbb{F}^n$ is a linear code that is ε-approximately (α, ℓ, L)-locally list recoverable with query complexity $Q \geq n'$. Suppose that $C' : \mathbb{F}^{k'} \to \mathbb{F}^{n'}$ is a linear code of relative distance δ' that is (α', ℓ, L')-globally list recoverable. Then $C \otimes C' : \mathbb{F}^{k \times k'} \to \mathbb{F}^{n \times n'}$ is $(d_0\varepsilon/\delta')$-approximately $(\alpha/s, \ell, L^{s \log L'})$-locally list recoverable with query complexity $Q \cdot (s \log L')$.

Moreover, if the approximately local list recovery algorithm for C has preprocessing time $T_{pre} \geq \log n'$ and running time T, and C' can be list recovered in time $T' \leq T$, then the approximately local list recovery algorithm for $C \otimes C'$ has preprocessing time $T_{pre} \cdot (s \log L') + L^{s \log L'}$ and running time $T \cdot (s \log L')$.

[3] As noted in Remark 2.9, the reason that we do not consider this all-codeword-symbol definition throughout the entire paper is that our reduction from approximately-local to local (Lemma 3.1) only works for our original message-symbol definition.

In Lemma 4.2, one should think of all of the "global" parameters (that is, $\delta', \alpha', \varepsilon, L'$ and s) as constants (or as slowly growing functions of n). In that case, if we start with a code that is ε-approximately (α, ℓ, L)-locally list recoverable with query complexity Q, then the final code would be $O(\varepsilon)$-approximately $(\Omega(\alpha), \ell, L^{O(1)})$-locally list recoverable with query complexity $O(Q)$.

We prove Lemma 4.2 in Section 4.1, but first we show how Lemma 4.1 follows from Lemma 4.2.

Proof of Lemma 4.1. The main idea is to start with the code C, and iteratively tensor with a new copy of C for $t-1$ times.

In more detail, let C be the globally list recoverable code guaranteed by the lemma statement. We observe that C is also $\left(\varepsilon \cdot (\delta/d_0)^{t-1}\right)$-approximately (α, ℓ, L) list recoverable with query complexity n in time $T_0(n) = \text{poly}(n)$, and with preprocessing time $P_0(n) = O(\max\{\log(n), L\})$, where d_0 is the constant from Lemma 4.2. Above, we have used the "trivial" approximately local list recovery algorithm that queries all the coordinates of C and globally list recovers C.

Choose the constant b_0 for the statement of Lemma 4.1 to be the same as the constant b_0 guaranteed by Lemma 4.2. With this choice of b_0, let

$$s_0 = (\delta\alpha\varepsilon)^{-b_0}$$

so that the requirement in the statement of Lemma 4.1 is that $s \geq s_0$.

A straightforward argument by induction shows that after we have applied Lemma 4.2 i times, we obtain a code $C^{\otimes(i+1)}$ which is $\left(\varepsilon \cdot (\delta/d_0)^{t-1-i}\right)$-approximately $(\alpha \cdot s_0^{-ti}, \ell, L^{s_0^{ti} \cdot \log^i(L)})$-locally list recoverable, with query complexity $n \cdot s_0^{ti} \log^i(L)$, and in time $T_0(n) \cdot s_0^{ti} \log^i(L)$, and with pre-processing time $P_0(n) \cdot s_0^{ti} \log^i(L) + L^{s_0^{ti} \log^i(L)}$. (Recall that $T_0(n)$ is the running time of the algorithm for C, and $P_0(n) = O(\max\{\log(n), L\})$ is the pre-processing time). We omit the details of the inductive step here for readability, but for completeness they can be found in Appendix C.

Thus, we conclude that $C^{\otimes t}$ is ε-approximately $(\alpha \cdot s_0^{-t^2}, \ell, L^{s_0^{t^2} \cdot \log^t(L)})$-locally list recoverable with query complexity $n \cdot {s_0}^{t^2} \log^t(L)$ and in time $\text{poly}(n) \cdot {s_0}^{t^2} \log^t(L)$, with pre-processing time $O\left(\log(n) \cdot s_0^{ti} \log^i(L) + L^{s_0^{ti} \log^i(L)}\right)$. In particular, for any $s \geq s_0$, the conclusion of Lemma 4.1 holds.

□

4.1 Proof of Main Technical Lemma 4.2

In this section we prove Lemma 4.2.

4.1.1 Approximate Local List Recovery Algorithm

Our goal is to find a randomized algorithm $\tilde{A}$ that outputs a list of (deterministic) local algorithms, and satisfies the following: For any codeword of $C \otimes C'$ that is consistent with most of the input lists, with high probability over the randomness of $\tilde{A}$, one of the local algorithms in the output list of $\tilde{A}$ correctly computes most of the codeword coordinates.

First, as per Remark 2.8, we assume that the ε-approximately (α, ℓ, L)-locally list recoverable code C which we are given in the statement of the lemma has a local list recovery algorithm A with failure probability ε, at the cost of increasing the output list size and the preprocessing time by a factor of $O(\log(1/\varepsilon))$. Thus, for some constant b_1, we assume that A produces a list of at most $b_1 L \log(1/\varepsilon)$ deterministic local algorithms, so that with probability at least $1-\varepsilon$, for all codewords

$c \in C$ consistent with the input lists, there is a local algorithm in the list which correctly computes a $1-\varepsilon$ fraction of the coordinates.

The algorithm $\tilde{A}$ produces the list of local algorithms as follows. It first chooses a random subset $J = \{j_1, \ldots, j_m\} \subseteq [n']$ of $m := b_2 s \log L'$ columns, where $b_2 = \Theta(1/\log(1/\varepsilon))$ will be chosen later. It then runs the approximately local list recovery algorithm A for C independently m times, one for each of the columns $j_1, \ldots, j_m$. (Notice that since the input to A is just 1^n, it does not matter at this stage which column we think of running it on). Let $\mathcal{L}_1, \ldots, \mathcal{L}_m$ denote the lists of local algorithms output by A in each of these runs. Finally, for every choice of local algorithms $A_1 \in \mathcal{L}_1, \ldots, A_m \in \mathcal{L}_m$, the algorithm $\tilde{A}$ outputs a local algorithm indexed by $(A_1, \ldots, A_m)$. The formal description of the algorithm $\tilde{A}$ appears in Algorithm 1.

Algorithm 1 The approximately local list recovery algorithm $\tilde{A}$ for $C \otimes C'$.

function $\tilde{A}$

▷ $\tilde{A}$ receives as input a parameter m.

Choose a random subset $J = \{j_1, \ldots, j_m\} \subseteq [n']$ of size m.

for $r = 1, \ldots, m$ **do**

Run the approximately local list recovery algorithm A for C, and let $\mathcal{L}_r$ be the list of local algorithms output by A.

end for

For any choice of local algorithms $A_1 \in \mathcal{L}_1, \ldots, A_m \in \mathcal{L}_m$, output a local algorithm $A_{(A_1,\ldots,A_m)}$.

end function

We now describe the local algorithm $A_{(A_1,\ldots,A_m)}$. Recall that the algorithm $A_{(A_1,\ldots,A_m)}$ is given as input a codeword coordinate $(i,j) \in [n] \times [n']$ in the tensor product code $C \otimes C'$, is allowed to query the input lists at every coordinate of $C \otimes C'$, and must produce a guess for the codeword value indexed by (i,j).

To this end, the algorithm $A_{(A_1,\ldots,A_m)}$ first uses the local algorithms $A_1, \ldots, A_m$ to obtain guesses for all positions in $\{i\} \times J$. Specifically, this is done by running on each column $j_r \in J$ the local algorithm A_r on input i and oracle access to the column j_r. Let v_r be the guess for the symbol at position (i, j_r) produced by A_r. At this point we have candidate symbols $(v_1, \ldots, v_m)$ for all positions in $\{i\} \times J$.

Next the algorithm $A_{(A_1,\ldots,A_m)}$ runs the global list recovery algorithm for C' on row i, and chooses a codeword c' from the output list $\mathcal{L}'$ that agrees the most with the candidate symbols $(v_1, \ldots, v_m)$ on J. Finally, the jth symbol of c' is the guess of the algorithm $A_{(A_1,\ldots,A_m)}$ for the (i,j) symbol of the tensor codeword. The formal description of the local algorithm $A_{(A_1,\ldots,A_m)}$ is given in Algorithm 2.

4.1.2 Output list size, query complexity, and running time

The output list size is number of local algorithms output by $\tilde{A}$ which is

$$(b_1 L \log(1/\varepsilon))^m = (b_1 L \log(1/\varepsilon))^{b_2 s \log L'} \leq L^{s \log L'},$$

as long as $L > 1$ and ε is sufficiently small, with a choice of $b_2 = \Theta(1/\log(1/\varepsilon))$.

The local algorithm $A_{(A_1,\ldots,A_m)}$ invokes a local algorithm A_j for C on m different columns, and the global algorithm for C' on a single row. Thus, the query complexity is $Q \cdot m + n'$ which is at

Algorithm 2 The local algorithm $A_{(A_1,\ldots,A_m)}$ for $C \otimes C'$.

function $A_{(A_1,\ldots,A_m)}((i,j) \in [n] \times [n'])$

▷ $A_{(A_1,\ldots,A_m)}$ receives oracle access to a matrix of lists $S \in \binom{\mathbb{F}}{\ell}^{n \times n'}$, and $J = \{j_1, \ldots, j_r\}$

for $r = 1, \ldots, m$ **do**

Run A_r on input i and oracle access to the j_r-th column $S|_{[n]\times\{j_r\}}$.

Let $v_r \leftarrow A_r(i)$.

▷ v_r is a candidate for the symbol at position $(i, j_r) \in [n] \times [n']$.

end for

▷ At this point, we have candidate symbols $(v_1, \ldots, v_m)$ for every position in $\{i\} \times J$.

Run the global list recovery algorithm for C' on the i-th row $S|_{\{i\}\times[n']}$, let $\mathcal{L}' \subseteq \mathbb{F}^{n'}$ denote the output list of codewords.

Choose a codeword $c' \in \mathcal{L}'$ such that $c'|_J$ is closest to $(v_1, \ldots, v_m)$ (breaking ties arbitrarily).

Return: c'_j

end function

most

$$Q \cdot m + n' \leq Q \cdot (m+1) = Q \cdot \left(b_2 s \log L' + 1\right) \leq Q \cdot s \log L',$$

using the assumption that $Q \geq n'$ and that $b_2 \leq 1/2$. By same reasoning as above, we also have that running time of the algorithm $A_{(A_1,\ldots,A_m)}$ is

$$T \cdot m + T' \leq T \cdot (m+1) = T \cdot \left(b_2 s \log L' + 1\right) \leq T \cdot s \log L'.$$

Finally, we bound the preprocessing time, which is the running time of the algorithm $\tilde{A}$. The algorithm $\tilde{A}$ can sample the set J in time $m \log n'$, then runs in time $T_{\text{pre}} \cdot b_1 \log(1/\varepsilon) \cdot m$ to generate the m output lists of A, and finally generates all output local algorithms in time $(b_1 L \log(1/\varepsilon))^m$. So the total preprocessing time is

$$\begin{aligned}
&m \log n' + T_{\text{pre}} \cdot b_1 \log(1/\varepsilon) \cdot m + (b_1 L \log(1/\varepsilon))^m \\
&\quad \leq (1+b_1) T_{\text{pre}} \log(1/\varepsilon) \cdot m + (b_1 L \log(1/\varepsilon))^m \\
&\quad = (1+b_1) T_{\text{pre}} \log(1/\varepsilon) \cdot (b_2 s \log L') + (b_1 L \log(1/\varepsilon))^{b_2 s \log L'} \\
&\quad \leq T_{\text{pre}} s \log L' + L^{s \log L'},
\end{aligned}$$

where the first inequality is by the assumption that $T_{\text{pre}} \geq \log n'$, and the second is by choosing $b_2 = \Theta(1/\log(1/\varepsilon))$ sufficiently small. Thus, the output list size, query complexity, and running time are all as desired.

4.1.3 Correctness

Let $\tilde{c}$ be a codeword of the tensor code $C \otimes C'$ that is consistent with all but an (α/s)-fraction of the input lists. Our goal is to show that with high probability (at least $2/3$) over the randomness of $\tilde{A}$, there exists a local algorithm $A_{(A_1,\ldots,A_m)}$ in the output list of $\tilde{A}$ that correctly computes all but a $O(\varepsilon/\delta')$-fraction of the coordinates of $\tilde{c}$.

For $r = 1, \ldots, m$, let A_r be the local algorithm in $\mathcal{L}_r$ that correctly decodes the largest number of coordinates of $\tilde{c}$ on the column j_r, *i.e.*, the local algorithm $A_r \in \mathcal{L}_r$ for which the set $\{i \in [n] \mid$

$A_r(i) = \tilde{c}_{i,j_r}\}$ is largest (breaking ties arbitrarily). We will show that with high probability over the randomness of $\tilde{A}$, the corresponding local algorithm $A_{(A_1,\ldots,A_m)}$ correctly computes all but a $O(\varepsilon/\delta')$-fraction of the coordinates of $\tilde{c}$.

To show the above, we claim that with high probability over the randomness of $\tilde{A}$, at least a $(1 - O(\varepsilon/\delta'))$-fraction of the rows of $\tilde{c}$ are "good," in the sense that the local algorithm $A_{(A_1,\ldots,A_m)}$ defined above will correctly decode *all* the coordinates of $\tilde{c}$ on these rows. More formally, we define a "good row" as follows.

Definition 4.3 (Good row)**.** A row $i \in [n]$ is good (with respect to $\tilde{c}$, J, and $A_1, \ldots A_m$) if it satisfies the following properties:

1. The codeword $\tilde{c}$ is consistent with all but an α'-fraction of the input lists on row i.
2. Let $\mathcal{L}' \subseteq \mathbb{F}^{n'}$ denote the list of all codewords in C' that are consistent with all but α'-fraction of the input lists on row i. Then $\mathrm{dist}(\mathcal{L}'|_J) > \delta'/2$. That is, for any pair of distinct codewords $c', c'' \in \mathcal{L}'$ it holds that $\mathrm{dist}(c'|_J, c''|_J) > \delta'/2$.
3. For $r = 1, \ldots, m$ let $v_r := A_r(i)$. Then $\mathrm{dist}\left(\tilde{c}|_{\{i\}\times J}, (v_1, \ldots, v_m)\right) \leq \delta'/4$.

Claim 4.4 below shows that the local algorithm $A_{(A_1,\ldots,A_m)}$ succeeds in correctly decoding *all* the coordinates of $\tilde{c}$ on a good row, while Claim 4.5 states that with probability at least $2/3$ over the randomness of $\tilde{A}$, at least a $(1 - O(\varepsilon/\delta'))$-fraction of the rows are good. The combination of these two claims then implies the desired conclusion.

Claim 4.4. *The local algorithm $A_{(A_1,\ldots,A_m)}$ correctly decodes all the coordinates of $\tilde{c}$ on a good row.*

Proof. Suppose that row i is good. By Property (1) in the definition of good, $\tilde{c}$ is consistent with all but an α'-fraction of the input lists on row i, and so $c' := \tilde{c}|_{\{i\}\times[n']}$, the restriction of $\tilde{c}$ to the i-th row, belongs to $\mathcal{L}'$. By Property (3) in the definition of good,

$$\mathrm{dist}\left(c'|_J, (v_1, \ldots, v_m)\right) \leq \delta'/4.$$

On the other hand, by Property (2) in the definition of good, and by the triangle inequality, for any other codeword $c'' \in \mathcal{L}'$ we have

$$\mathrm{dist}\left(c''|_J, (v_1, \ldots, v_m)\right) \geq \mathrm{dist}(c'|_J, c''|_J) - \mathrm{dist}\left(c'|_J, (v_1, \ldots, v_m)\right) > \delta'/4.$$

Thus the local algorithm $A_{(A_1,\ldots,A_m)}$ will choose the codeword $c' = \tilde{c}|_{\{i\}\times[n']}$ on the i-th row, and consequently all its decodings on the i-th row will be consistent with $\tilde{c}$.

□

Claim 4.5. *With probability at least $2/3$ over the randomness of $\tilde{A}$, at least a $(1-O(\varepsilon/\delta'))$-fraction of the rows are good.*

For the proof of the above claim we shall also use the notion of a "good column," which we define below as a column $j_r \in J$ on which most of the coordinates of $\tilde{c}$ are decoded correctly by the local algorithm A_r.

Definition 4.6 (Good column)**.** A column $j_r \in J$ is good if the local algorithm A_r correctly decodes all but an ε-fraction of the coordinates of $\tilde{c}$ on column j_r.

Once more, we shall show that with high probability over the randomness of $\tilde{A}$, a large fraction of the columns in J are good.

Claim 4.7. *With probability at least* 0.9 *over the randomness of* $\tilde{A}$*, at least a* $(1 - O(\varepsilon))$*-fraction of the columns in* J *are good.*

Proof. We first claim that with probability at least 0.99 over the randomness of $\tilde{A}$, for at least a $(1 - 2\varepsilon)$-fraction of the columns $j_r \in J$ it holds that $\tilde{c}$ is consistent with all but α-fraction of the input lists on column j_r. To see this note first that by assumption $\tilde{c}$ agrees with all but a $\alpha/s \leq \alpha \cdot \varepsilon$ of the input lists, where the inequality follows from the definition of s and by taking $b_0 \geq 1$. Thus, by averaging, for at least a $(1 - \varepsilon)$-fraction of the columns $j \in [n']$ it holds that $\tilde{c}$ is consistent with all but an α-fraction of the input lists on column j. By a Chernoff bound (see Theorem D.2 in the appendix), this implies in turn that with probability at least

$$1 - \exp\big(-m \cdot \varepsilon^2\big) \geq 0.99$$

over the choice of J, for at least a $(1-2\varepsilon)$-fraction of the columns $j_r \in J$ it holds that $\tilde{c}$ is consistent with all but an α-fraction of the input lists on column j_r. In the inequalty above, we are using the choice

$$m = \Theta(\frac{s \log L'}{\log(1/\varepsilon)} = \Omega\left(\frac{1}{\log(1/\varepsilon)}\left(\frac{1}{\varepsilon\delta'\alpha'}\right)^{b_0} \log L'\right),$$

along with a sufficiently large choice of the constant b_0.

As noted at the beginning of the proof, we are considering an amplified version of the approximate local list recovery algorithm A for C, so that the failure probability is at most ε. By Hoeffding's inequality (Theorem D.1), this implies that with probability at least $1 - \exp(-m \cdot \varepsilon^2) \geq 0.99$, the algorithm A succeeds in at least $(1 - 2\varepsilon)$-fraction of the invocations. Above, the inequality follows as before because of the choice of s and a sufficiently large choice of b_0.

By a union bound, we conclude that with probability at least 0.9 over the randomness of $\tilde{A}$, for at least a $(1 - 4\varepsilon)$-fraction of the columns $j_r \in J$ it holds that both $\tilde{c}$ is consistent with all but α-fraction of the input lists on column j_r, and A does not fail on this column: that is, there is a local algorithm returned by A which correctly computes at least a $1 - \varepsilon$ fraction of the coordinates of column j_r. For such a column j_r, since A_r is chosen to be the local algorithm returned by A which correctly computes the largest fraction of coordinates of $\tilde{c}$ on the column j_r, we conclude that the column j_r is good. All together, this shows that at least a $(1 - 4\varepsilon)$-fraction of the columns $j_r \in J$ are good. □

We now proceed to the proof of Claim 4.5.

Proof of Claim 4.5. We will show that each of the three properties in the definition of a good row holds for at least $(1 - O(\varepsilon/\delta'))$-fraction of the rows, with probability at least 0.9 over the randomness of $\tilde{A}$. Then the claim will follow by a union bound over the fraction of bad rows and error probability of $\tilde{A}$.

Property (1): By assumption $\tilde{c}$ agrees with all but an $\alpha/s \leq \alpha' \cdot \varepsilon$ of the input lists, where the inequality follows from the definition of s and an assumption that $b_0 \geq 1$. By an averaging arguement, for at least a $(1 - \varepsilon)$-fraction of the rows $i \in [n]$ it holds that $\tilde{c}$ is consistent with all but an α'-fraction of the input lists on row i. So at least a $(1 - \varepsilon)$-fraction of the rows satisfy Property (1) (regardless of the randomness of $\tilde{A}$).

Property (2): By assumption, C' has relative distance at least δ', and so $\mathrm{dist}(c', c'') \geq \delta'$ for any pair of codewords $c', c'' \in \mathcal{L}'$. A Chernoff bound (Theorem D.2) then implies that with probability at least

$$1 - \exp\big(-m \cdot (\delta')^2\big) = 1 - \exp\big(-b_2 s \log L' \cdot (\delta')^2\big) \geq 1 - \frac{0.1\varepsilon}{(L')^2}$$

over the choice of J it holds that $\mathrm{dist}(c'|_J, c''|_J) > \delta'/2$. Above, the inequality follows by choosing the constant b_0 sufficiently large and $b_2 = \Theta(1/\log(1/\varepsilon))$. By a union bound over all $(L')^2$ pairs of elements in $\mathcal{L}'$, this implies in turn that $\mathrm{dist}(\mathcal{L}'|_J) > \delta'/2$ with probability at least $1 - 0.1\varepsilon$ over the choice of J. Finally, by an averaging argument we conclude that with probability at least 0.9 over the choice of J (and so also over the randomness of $\tilde{A}$), at least a $(1-\varepsilon)$-fraction of the rows satisfy Property (2).

Property (3): By Claim 4.7, with probability at least 0.9 over the randomness of $\tilde{A}$, at least a $(1-O(\varepsilon))$-fraction of the columns in J are good, where in a good column $j_r \in J$ all but an ε-fraction of the coordinates of $\tilde{c}$ are decoded correctly by A_r. This implies in turn that the fraction of points $(i, j_r) \in [n] \times J$ on which $A_r(i) \neq \tilde{c}_{i,j_r}$ is at most $O(\varepsilon)$, and so at least a $(1-O(\varepsilon/\delta'))$-fraction of the rows i have at most $(\delta'/4)$-fraction of entries $j_r \in J$ for which $A_r(i) \neq \tilde{c}_{i,j_r}$. Thus with probability at least 0.9 over the randomness of $\tilde{A}$, at least a $(1-O(\varepsilon/\delta'))$-fraction of the rows satisfy Property (3).

Together, the three paragraphs above imply that at least a $1 - O(\varepsilon) - O(\varepsilon) - O(\varepsilon/\delta')$ fraction of the rows satisfy all three properties, which proves the claim. □

5 Instantiations

We conclude by instantiating our main "global to approximately-local" transformation (Lemma 4.1), followed by the basic list recovery transformations (Lemmas 3.1, 3.2, and 3.4), with known constructions of high-rate globally list recoverable codes to obtain capacity-achieving locally list recoverable codes, which proves our main Theorems 1.2, 1.3, and 1.4.

5.1 Instantiating with a random linear code: Proof of Theorem 1.2

We start by instantiating our transformations with the random linear code given by Corollary 2.2, restated below.

Corollary (Corollary 2.2, restated)**.** *For any $\rho \in [0,1]$, $\varepsilon > 0$, $\ell \geq 1$, and for sufficiently large prime power q and integer n, a random linear code $C : \mathbb{F}_q^k \to \mathbb{F}_q^n$ of rate ρ has relative distance at least $1 - \rho - \varepsilon$, and is $(1 - \rho - \varepsilon, \ell, q^{O(\ell/\varepsilon)})$-list recoverable, with probability at least $1 - \exp(-n)$.*

Using the above corollary we obtain the following lemma which implies Theorem 1.2.

Lemma 5.1. *For any constants $\rho \in [0,1]$, $\varepsilon, \beta > 0$, and $\ell \geq 1$ there exist integers σ, L, and an infinite family of codes $\{C_n\}_n$ that satisfy the following.*

- *C_n is an $\mathbb{F}_2$-linear code of block length n and alphabet size σ.*
- *C_n has rate ρ and relative distance at least $1 - \rho - \varepsilon$.*
- *C_n is $(1 - \rho - \varepsilon, \ell, L)$-locally list recoverable with query complexity n^β.*

Proof. The proof outline is as follows. We start with the high-rate globally list recoverable code C given by Corollary 2.2, and use Lemma 4.1 to turn C into a high-rate approximately locally list recoverable code C' by raising C to a sufficiently large tensor power. We then use Lemma 3.1 to turn C' into a high-rate locally list recoverable code C'' by pre-encoding C' with a high-rate locally decodable code. Finally, we use the AEL transformation (Lemma 3.2) to turn C'' into a capacity-achieving locally list recoverable code $\tilde{C}$.

Below, we focus in more detail on each of the codes described above.

High-rate globally list recoverable code C: The initial code C will be the high-rate globally list recoverable code given by Corollary 2.2. Specifically, we choose the block length of C to be $n^{\beta/2}$, smaller than the final desired query complexity, and the rate to be high, at least $1-\varepsilon\beta/16$.

As we will see in a moment, the rationale for these choices is that if we raise C to the tensor power of $2/\beta$, Lemma 4.1 will yield a code of block length n and query complexity smaller than n^{β}, with rate at least $1-\varepsilon$.

Note that Corollary 2.2 guarantees the existence of a code C with this block length and rate that has relative distance $\Omega(1)$, and is $(\Omega(1), \ell', O(1))$-globally list recoverable for any constant $\ell' \geq 1$, provided that the alphabet size is a sufficiently large constant prime power, and n is sufficiently large.

High-rate approximately locally list recoverable code C': Let $C' = C^{\otimes(2/\beta)}$ be the code obtained by taking the $(2/\beta)$-th tensor power of C. Then C' has block length n, and by Lemma 4.1, it is $(\varepsilon/100)$-approximately $(\Omega(1), \ell', O(1))$-locally list recoverable with query complexity $O(n^{\beta/2})$ for any constant $\varepsilon > 0$ and $\ell' \geq 1$. Moreover, by Corollary 2.13 C' has rate $(1-\varepsilon\beta/16)^{2/\beta} \geq 1-\varepsilon/8$ and relative distance $\Omega(1)$.

High-rate locally list recoverable code C'': We obtain C'' by pre-encoding C' with a high-rate locally decodable code D' that is guaranteed by Theorem 2.5. Specifically, to satisfy the conditions of Lemma 3.1, we choose the block length of D' to be equal to the message length of C' which is $(1-\varepsilon/8)n$, and the decoding radius of D' to be equal to $\varepsilon/100$. Since the rate of C'' is the product of the rates of C' and D', we also require that the rate of D' is high, specifically $1-\varepsilon/8$. Note that Theorem 2.5 guarantees the existence of such a code D' with query complexity $n^{o(1)}$ for infinitely many values of n, and for a constant alphabet size that is a power of 2. Moreover, the alphabet size can be increased to any arbitrarily large constant by grouping together consecutive symbols and noting that this does not effect the asymptotic behavior of the code in the parameter regime that we are operating in.

Lemma 3.1 then implies that C'' is a code of block length n that is $(\Omega(1), \ell', O(1))$-locally list recoverable with query complexity $n^{\beta/2+o(1)}$ for any constant $\ell' \geq 1$. Note also that the rate of C'' is the product of the rates of C' and D', and so is at least $1-\varepsilon/4$, while the relative distance of C'' is the same as that of C', and so is constant $\Omega(1)$.

Capacity-achieving locally list recoverable code $\tilde{C}$: Finally, we obtain $\tilde{C}$ by applying the AEL transformation (Lemma 3.2) with the outer code being the code C'' constructed so far, and the inner code being a capacity-achieving globally list recoverable code D'' given by Corollary 2.2. Specifically, as we would like the final code $\tilde{C}$ to have rate ρ and relative distance and list recovery

radius $1-\rho-\varepsilon$, and as C'' has rate $1-\varepsilon/4$, we require that D'' has rate $\rho+\varepsilon/4$ and relative distance and list recovery radius $1-\rho-\varepsilon/2$.

Note that Corollary 2.2 guarantees the existence of a code D'' as above that is $(1-\rho-\varepsilon/2, \ell, \ell')$-globally list recoverable for sufficiently large constant ℓ', provided that the alphabet size is a sufficiently large constant prime power, and the block length is a sufficiently large constant. To satisfy the conditions of Lemma 3.2, we further require that the block length of D'' is a sufficiently large constant, as required by the lemma, and that the alphabet size of C'' equals the domain size of D''.

Lemma 3.2 then implies that $\tilde{C}$ is a code of block length n that is $(1-\rho-\varepsilon, \ell, O(1))$-locally list recoverable with query complexity $n^{\beta/2+o(1)} \ll n^{\beta}$. Moreover, the code $\tilde{C}$ has rate ρ, relative distance at least $1-\rho-\varepsilon$, and constant alphabet size.

Finally, note that all codes in the process can be taken to be $\mathbb{F}_2$-linear, and all transformations preserve $\mathbb{F}_2$-linearity, so the final code can be guaranteed to be $\mathbb{F}_2$-linear as well. □

5.2 Instantiating with an AG code for polynomial query complexity: Proof of Theorem 1.3

Next we instantiate our transformations with the algebraic geometry subcodes of [GX13, GK16b] to obtain the following lemma which implies our main Theorem 1.3.

Lemma 5.2. *For any constants $\rho \in [0,1]$, $\varepsilon, \beta > 0$, and $\ell \geq 1$ there exist an integer σ, and an infinite family of codes $\{C_n\}_n$ that satisfy the following.*

- *C_n is an $\mathbb{F}_2$-linear code of block length n and alphabet size σ.*
- *C_n has rate ρ and relative distance at least $1-\rho-\varepsilon$.*
- *C_n is $(1-\rho-\varepsilon, \ell, L)$-locally list recoverable with query complexity n^{β} for $L = \exp\exp\exp(\log^* n)$ with preprocessing time $(\log n)^{1+o(1)}$ and running time $n^{O(\beta)}$.*
- *C_n is $(1-\rho-\varepsilon, \ell, L)$-globally list recoverable for $L = \exp\exp\exp(\log^* n)$ in time $n^{1+O(\beta)}$.*
- *C_n is encodable in time $n^{1+O(\beta)}$.*

To prove the above lemma we use the algebraic geometry subcodes of [GX13, GK16b]. However, we cannot quite use these codes as a black box, for two reasons. First, the analysis in [GX13] only establishes list-*decodability*, rather than list recoverability. Fortunately, list recoverability follows from exactly the same argument as list-decodability. Second, these codes are linear over a subfield, but are not themselves linear, while our arguments require linearity over the whole alphabet. Fortunately, we can achieve the appropriate linearity by concatenating the AG subcode with a small high-rate globally list recoverable linear code, which exists by Corollary 2.2. We handle these modifications to the approach of [GX13, GK16b] in Appendix A, and the final properties we need are summarized in Theorem A.1, which we prove in the Appendix and restate below.

Theorem (Theorem A.1, restated)**.** *There exists an absolute constant b_0 so that the following holds. For any $\varepsilon > 0$, $\ell \geq 1$, $q \geq \ell^{b_0/\varepsilon}$ that is an even power of a prime[4], and integer $n \geq q^{b_0 \ell/\varepsilon}$, there exists a linear code $C : \mathbb{F}_q^k \to \mathbb{F}_q^n$ of rate $1-\varepsilon$ and relative distance $\Omega(\varepsilon^2)$ that is $(\Omega(\varepsilon^2), \ell, L)$-list recoverable for $L = q^{q^{(\ell/\varepsilon)\cdot\exp(\log^* n)}}$. Moreover, C can be encoded in time $\mathrm{poly}(n, \log q)$ and list recovered in time $\mathrm{poly}(n, L)$.*

[4]That is, q is of the form p^{2t} for a prime p and for an integer t.

Next we prove Lemma 5.2 based on the above theorem.

Proof of Lemma 5.2. The proof is identical to that of Lemma 5.1, replacing the initial random linear code C given by Corollary 2.2 with the AG code given by Theorem A.1. It can be verified that the proof of Lemma 5.1 goes through, when increasing the output list size L from constant to $\exp\exp\exp(\log^* n)$.

As for the running times, note that by Theorem A.1, C can be encoded and list recovered in time $n^{O(\beta)}$. Lemma 4.1 then implies that the approximately local list recovery algorithm for C' has preprocessing time $(\log n)^{1+o(1)}$ and running time $n^{O(\beta)}$, while by Corollary 2.13, the encoding time of C' is $n^{1+O(\beta)}$. Lemmas 3.1 and 3.2 imply in turn that the same holds for C'' and $\tilde{C}$, and by Lemma 3.4, the code $\tilde{C}$ is also globally list recoverable in time $n^{1+O(\beta)}$.

□

5.3 Instantiating with an AG code for sub-polynomial query complexity: Proof of Theorem 1.4

Our final instantiation uses once more the algebraic geometry codes given by Theorem A.1, but with a different choice of parameters, which gives the following lemma that implies our main Theorem 1.4.

Lemma 5.3. *For any constants $\rho \in [0,1]$, $\varepsilon > 0$, and $\ell \geq 1$ there exists an infinite family of codes $\{C_n\}_n$ that satisfy the following.*

- *C_n is an $\mathbb{F}_2$-linear code of block length n and alphabet size $n^{o(1)}$.*
- *C_n has rate ρ and relative distance at least $1-\rho-\varepsilon$.*
- *C_n is $(1-\rho-\varepsilon, \ell, n^{o(1)})$-locally list recoverable with query complexity, preprocessing time, and running time $n^{o(1)}$.*
- *C_n is $(1-\rho-\varepsilon, \ell, n^{o(1)})$-globally list recoverable in time $n^{1+o(1)}$.*
- *C_n is encodable in time $n^{1+o(1)}$.*

Proof. The proof is identical to that of Lemmas 5.1 and 5.2, taking β to be slightly subconstant, specifically $\beta := (\log\log n)^{-o(1)}$ (where the $o(1)$ term in the exponent is an arbitrarily slowly decreasing function of n). As setting of parameters is slightly different from the previous lemmas, we provide a complete proof below.

High-rate globally list recoverable code C: As in the proof of Lemma 5.1, we choose the initial code C to be a code of block length $n^{\beta/2} = n^{o(1)}$ and rate $1-\varepsilon\beta/16 = 1-(\log\log n)^{-o(1)}$. Theorem A.1 guarantees the existence of such a code C that has relative distance $(\log\log n)^{-o(1)}$, and is $((\log\log n)^{-o(1)}, \ell', \exp\exp((\log\log n)^{o(1)}))$-globally list recoverable for any constant $\ell' \geq 1$, provided that the alphabet size is sufficiently large even power of a prime $\exp((\log\log n)^{o(1)})$. Moreover, C can be encoded and list recovered in time $n^{O(\beta)} = n^{o(1)}$.

High-rate approximately locally list recoverable code C': As before, let C' be the code obtained by taking C to a tensor power of $2/\beta = (\log\log n)^{o(1)}$. Then C' has block length n, rate at least $1-\varepsilon/8$, relative distance $\exp(-(\log\log n)^{o(1)})$, and by Lemma 4.1, it is $(\varepsilon/100)$-approximately $(\exp(-(\log\log n)^{o(1)}), \ell', n^{o(1)})$-locally list recoverable with query complexity $n^{o(1)}$ for any constants $\varepsilon > 0$ and $\ell' \geq 1$. Moreover, the approximately local list recovery algorithm for C' has preprocessing and running time $n^{o(1)}$, while by Corollary 2.13, the encoding time of C' is $n^{1+o(1)}$.

High-rate locally list recoverable code C'': Once more, we obtain C'' by pre-encoding C' with a high-rate locally decodable code D' of block length $(1-\varepsilon/8)n$, rate $1-\varepsilon/8$, and decoding radius $\varepsilon/100$. Theorem 2.5 guarantees the existence of such a code D' with query complexity $n^{o(1)}$ for infinite values of n, and with constant alphabet size that is a power of 2. Moreover, the alphabet size can be increased to $\exp((\log\log n)^{o(1)})$—the alphabet size of C'—by grouping together consecutive symbols, and noting that this does not effect the asymptotic behavior of the code.

Then C'' is a code of block length n, rate $1-\varepsilon/4$, and relative distance $\exp(-(\log\log n)^{o(1)})$, and by Lemma 3.1, it is $(\exp(-(\log\log n)^{o(1)}), \ell', n^{o(1)})$-locally list recoverable with query complexity $n^{o(1)}$ for any constant $\ell' \geq 1$. Moreover, the local list recovery algorithm for C'' has preprocessing and running time $n^{o(1)}$, and since the encoding time of D' is $n^{1+o(1)}$, the encoding time of C'' is $n^{1+o(1)}$ as well.

Capacity-achieving locally list recoverable code $\tilde{C}$: As before, $\tilde{C}$ is obtained by applying the AEL transformation (Lemma 3.2) with the outer code being the code C'' constructed so far, and the inner code being a capacity-achieving globally list recoverable code D'' of rate $\rho+\varepsilon/4$ and relative distance and list recovery radius $1-\rho-\varepsilon/2$.

Corollary 2.2 guarantees the existence of code D'' as above that is $(1-\rho-\varepsilon/2, \ell, \ell')$-globally list recoverable for sufficiently large constant ℓ', provided that the alphabet size is a sufficiently large constant prime power, and the block length is a sufficiently large constant. To satisfy the conditions of Lemma 3.2, we further require that the block length of D'' is sufficiently large $\exp((\log\log n)^{o(1)})$, and that the alphabet size of C'' is $\exp\exp((\log\log n)^{o(1)})$—the domain size of D''—which can be achieved by grouping together consecutive symbols of C''.

Lemma 3.2 then implies that $\tilde{C}$ is a code of block length n, alphabet size $n^{o(1)}$, rate ρ, and relative distance $1-\rho-\varepsilon$, that is $(1-\rho-\varepsilon, \ell, n^{o(1)})$-locally list recoverable with query complexity $n^{o(1)}$. Moreover, the local list recovery algorithm for $\tilde{C}$ has preprocessing and running time $n^{o(1)}$, and the encoding time of $\tilde{C}$ is $n^{1+o(1)}$. Finally, Lemma 3.4 implies that the code $\tilde{C}$ is also globally list recoverable in time $n^{1+o(1)}$.

Once more we note that all codes in the process can be taken to be $\mathbb{F}_2$-linear, and all transformations preserve $\mathbb{F}_2$-linearity, so the final code can be guaranteed to be $\mathbb{F}_2$-linear as well. □

Acknowledgements. NRZ thanks Swastik Kopparty for raising the question of obtaining capacity-achieving locally list decodable codes, and Sivakanth Gopi, Swastik Kopparty, Rafael Oliveira and Shubhangi Saraf for many discussions on this topics. The current collaboration began during the Algorithmic Coding Theory Workshop at ICERM, we thank ICERM for their hospitality. BH was supported in part by NSF grant CNS-1513671, MW was supported in part by NSF grant CCF-1657049, MW and NRZ were supported in part by NSF-BSF grant CCF-1814629 and 2017732.

References

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

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

[BEAT] Avraham Ben-Aroya, Klim Efremenko, and journal = Amnon Ta-Shma. A note on amplifying the error-tolerance of locally decodable codes.

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

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

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

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

[Cox18] Nicholas Coxon. Fast systematic encoding of multiplicity codes. *Journal of Symbolic Computation*, 2018.

[CR05] Don Coppersmith and Atri Rudra. On the robust testability of tensor products of codes. ECCC TR05-104, 2005.

[DL12] Zeev Dvir and Shachar Lovett. Subspace evasive sets. In *Proceedings of the 44th Annual ACM Symposium on Theory of Computing (STOC)*, pages 351–358. ACM Press, 2012.

[DSW06] Irit Dinur, Madhu Sudan, and Avi Wigderson. Robust local testability of tensor products of LDPC codes. In *proceedings of the 9th International Workshop on Randomization and Computation (RANDOM)*, pages 304–315. Springer, 2006.

[GGR11] Parikshit Gopalan, Venkatesan Guruswami, and Prasad Raghavendra. List decoding tensor products and interleaved codes. *SIAM Journal on Computing*, 40(5):1432–1462, 2011.

[GI01] Venkatesan Guruswami and Piotr Indyk. Expander-based constructions of efficiently decodable codes. In *Proceedings of the 42nd Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 658–667. IEEE Computer Society, 2001.

[GI02] Venkatesan Guruswami and Piotr Indyk. Near-optimal linear-time codes for unique decoding and new list-decodable codes over smaller alphabets. In *Proceedings of the 34th Annual ACM Symposium on Theory of Computing (STOC)*, pages 812–821. ACM Press, 2002.

[GI03] Venkatesan Guruswami and Piotr Indyk. Linear time encodable and list decodable codes. In *Proceedings of the 35th Annual ACM Symposium on Theory of Computing (STOC)*, pages 812–821. ACM Press, 2003.

[GI04] Venkatesan Guruswami and Piotr Indyk. Efficiently decodable codes meeting gilbert-varshamov bound for low rates. In *Proceedings of the 15th Annual ACM-SIAM Symposium on Discrete Algorithm (SODA)*, pages 756–757. SIAM, 2004.

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

[GK16a] Alan Guo and Swastik Kopparty. List-decoding algorithms for lifted codes. *IEEE Transactions on Information Theory*, 62(5):2719 – 2725, 2016.

[GK16b] Venkatesan Guruswami and Swastik Kopparty. Explicit subspace designs. *Combinatorica*, 36(2):161–185, 2016.

[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.

[GL89] Oded Goldreich and Leonid A Levin. A hard-core predicate for all one-way functions. In *Proceedings of the 21st annual ACM symposium on Theory of computing (STOC)*, pages 25–32. ACM Press, 1989.

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

[GNP+13] Anna Gilbert, Hung Ngo, Ely Porat, Atri Rudra, and Martin Strauss. $\ell 2/\ell 2$-foreach sparse recovery with low risk. In *proceedings of the 40th International Colloquium on Automata, Languages, and Programming (ICALP)*, volume 7965 of *Lecture Notes in Computer Science*, pages 461–472. Springer, 2013.

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

[GS99] Venkatesan Guruswami and Madhu Sudan. Improved decoding of Reed-Solomon and algebraic-geometry codes. *IEEE Transactions on Information Theory*, 45(6), 1999.

[Gur01] Venkatesan Guruswami. *List decoding of error-correcting codes*. PhD thesis, MIT, 2001.

[Gur10] Venkatesan Guruswami. Cyclotomic function fields, artin–frobenius automorphisms, and list error correction with optimal rate. *Algebra & Number Theory*, 4(4):433–463, 2010.

[GUV09] Venkatesan Guruswami, Christopher Umans, and Salil Vadhan. Unbalanced expanders and randomness extractors from parvaresh-vardy codes. *Journal of the ACM*, 56(4):20:1–20:34, 2009.

[GW13] Venkatesan Guruswami and Carol Wang. Linear-algebraic list decoding for variants of reed-solomon codes. *IEEE Transactions on Information Theory*, 59(6):3257–3268, 2013.

[GX12a] Venkatesan Guruswami and Chaoping Xing. Folded codes from function field towers and improved optimal rate list decoding. In *Proceedings of the 44th annual ACM symposium on Theory of computing (STOC)*, pages 339–350. ACM, 2012.

[GX12b] Venkatesan Guruswami and Chaoping Xing. List decoding reed-solomon, algebraic-geometric, and gabidulin subcodes up to the singleton bound. *Electronic Colloquium on Computational Complexity (ECCC)*, 2012.

[GX13] Venkatesan Guruswami and Chaoping Xing. List decoding Reed-Solomon, Algebraic-Geometric, and Gabidulin subcodes up to the Singleton bound. In *Proceedings of the 45th annual ACM symposium on Theory of Computing (STOC)*, pages 843–852. ACM, 2013.

[GX14] Venkatesan Guruswami and Chaoping Xing. Optimal rate list decoding of folded algebraic-geometric codes over constant-sized alphabets. In *Proceedings of the 25th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1858–1866. SIAM, 2014.

[HIOS15] Iftach Haitner, Yuval Ishai, Eran Omri, and Ronen Shaltiel. Parallel hashing via list recoverability. In *proceedings of the 35th International Cryptology Conference (CRYPTO)*, volume 9216 of *Lecture Notes in Computer Science*, pages 173–190. Springer, 2015.

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

[HW18] Brett Hemenway and Mary Wootters. Linear-time list recovery of high-rate expander codes. *Information and Computation*, 261:202–218, 2018.

[INR10] Piotr Indyk, Hung Ngo, and Atri Rudra. Efficiently decodable non-adaptive group testing. In *Proceedings of the 21st Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1126–1142. SIAM, 2010.

[KM93] Eyal Kushilevitz and Yishay Mansour. Learning decision trees using the fourier spectrum. *SIAM Journal on Computing*, 22(6):1331–1348, 1993.

[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 ACM*, 64(2):11:1–11:42, 2017.

[Kop15] Swastik Kopparty. List-decoding multiplicity codes. *Theory Of Computing*, 11(5):149–182, 2015.

[KRSW18] Swastik Kopparty, Noga Ron-Zewi, Shubhangi Saraf, and Mary Wootters. Improved list decoding of folded reed-solomon and multiplicity codes. In *Proceedings of the 59th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*. IEEE Computer Society, 2018.

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

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

[MV05] Peter Bro Miltersen and N Variyam Vinodchandran. Derandomizing arthur–merlin games using hitting sets. *Computational Complexity*, 14(3):256–279, 2005.

[NPR12] Hung Ngo, Ely Porat, and Atri Rudra. Efficiently Decodable Compressed Sensing by List-Recoverable Codes and Recursion. In *proceedings of the 29th International Symposium on Theoretical Aspects of Computer Science (STACS)*, volume 14, pages 230–241, Dagstuhl, 2012.

[Sti09] Henning Stichtenoth. *Algebraic function fields and codes*, volume 254. Springer Science & Business Media, 2009.

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

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

[Tre01] Luca Trevisan. Extractors and pseudorandom generators. *Journal of the ACM*, 48(4):860–879, 2001.

[TZ04] Amnon Ta-Shma and David Zuckerman. Extractor codes. *IEEE Transactions on Information Theory*, 50(12):3015–3025, 2004.

[TZS06] Amnon Ta-Shma, David Zuckerman, and Shmuel Safra. Extractors from reed-muller codes. *Journal of Computer and System Sciences*, 72(5):786–812, 2006.

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

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

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

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

A List recovery of algebraic geometry codes

In this appendix, we outline how the approach of [GX13] needs to be changed in order to obtain linear list recoverable codes. The main theorem is as follows.

Theorem A.1. *There exists an absolute constant b_0 so that the following holds. For any $\varepsilon > 0$, $\ell \geq 1$, $q \geq \ell^{b_0/\varepsilon}$ that is an even power of a prime*[5]*, and integer $n \geq q^{b_0 \ell/\varepsilon}$, there exists a linear code $C : \mathbb{F}_q^k \to \mathbb{F}_q^n$ of rate $1 - \varepsilon$ and relative distance $\Omega(\varepsilon^2)$ that is $(\Omega(\varepsilon^2), \ell, L)$-list recoverable for $L = q^{q^{(\ell/\varepsilon)\cdot \exp(\log^* n)}}$. Moreover, C can be encoded in time $\mathrm{poly}(n, \log q)$ and list recovered in time $\mathrm{poly}(n, L)$.*

We remark that when ε, ℓ, q are constant the output list size L is $\exp\exp\exp(\log^* n)$ which is very slowly growing (although admittedly with extremely large constants).

We follow the approach of [GX13, GK16b]. In [GX13], Guruswami and Xing show how to construct high-rate list decodable codes over a constant alphabet, modulo a construction of *explicit subspace designs*. In [GK16b], Guruswami and Kopparty gave such constructions and used them to construct high-rate list decodable codes over constant-sized alphabets with small list sizes. We would like to use these codes here. However, there are two things which must be modified. First, the guarantees of [GX13, GK16b] are for list decodability, and we are after list recoverability. Fortunately, this follows from a standard modification of the techniques that they use. Second, the codes that they obtain are not linear, but rather are linear over a subfield of the alphabet. To correct this, we concatenate these codes with list recoverable linear codes of a constant length. A random linear code has this property, and since we only require them to be of constant length, we may find such a code, and run list recovery algorithms on it, in constant time.

We begin by addressing the leap from list decodability to list recovery, and then discuss the code concatenation step. We refer the reader to [GX13, GK16b] for the details (and, indeed, for several definitions); here we just outline the parts which are important for list recovery. The basic outline of the construction and the argument is as follows:

Step 1. Show that AG codes are list decodable, with large but very structured lists. We will extend this to list recoverability with structured lists.

Step 2. Show that one can efficiently find a subcode of the AG code which will avoid this sort of structure: this reduces the list size. This part of the argument goes through unchanged, and will yield a list recoverable code over $\mathbb{F}_{q^m}$ with small list size.

Once we have $\mathbb{F}_q$-linear codes over $\mathbb{F}_q^m$ that are list recoverable, we discuss the third step:

Step 3. The code produced is $\mathbb{F}_q$-linear (rather than $\mathbb{F}_{q^m}$-linear). This was fine for [GX13, GK16b], but we require a code which is linear over the alphabet it is defined over. To this end we concatenate the codes above with a random linear code of length m over $\mathbb{F}_q$. This will result in an $\mathbb{F}_q$-linear code over $\mathbb{F}_q$ that is list recoverable with small list sizes.

We briefly go through the details. First we give a short refresher/introduction to the notation. Then we handle the three steps above, in order. We note that throughout this appendix we will refer to Theorem and Lemma numbers in the extended version [GX12b] rather than the conference version [GX13].

Step 0. Algebraic Geometry Codes and basic notation. Since we do not need to open up the AG code machinery very much in order to extend the results of [GX13] to list recovery, we do not go into great detail here, and we refer the reader to [GX13] and the references therein for the

[5] That is, q is of the form p^{2t} for a prime p and for an integer t.

technical details, and to [Sti09] for a comprehensive treatment of AG codes. However, for the ease of exposition here (for the reader unfamiliar with AG codes), we will introduce some notation and explain the intuitive definitions of these notions. In particular, we will use the running example of a rational function field. We stress that this is *not* the final function field used; thus the intuition should be taken as intuition only.

Let $F/\mathbb{F}_q$ be a function field of genus g. One example, which may be helpful to keep in mind, of a genus 0 function field is the rational function field $\mathbb{F}_q(X)/\mathbb{F}_q$, which may be thought of as rational functions $f(X)/g(X)$, where $f, g \in \mathbb{F}_q[X]$ are irreducible polynomials. For the code construction, we will use a function field of larger genus (given by the Garcia-Stichtenoth tower, as in [GX13]), but we will use this example to intuitively define the algebraic objects that we need.

Let $P_\infty, P_1, \ldots, P_n$ be $n+1$ distinct $\mathbb{F}_q$-rational places (that is, of degree 1). Formally, these are ideals, but they are in one-to-one correspondence with $\mathbb{F}_q \cup \{\infty\}$, and let us think of them that way. For each such place P, there is a map (the *residue class map* with respect to P) which maps $F/\mathbb{F}_q$ to $\mathbb{F}_q$; we may think of this as function evaluation, and in our example of $\mathbb{F}_q(X)/\mathbb{F}_q$, if P is a place associated with a point $\alpha \in \mathbb{F}_q$, then indeed this maps $f(X)/g(X)$ to $f(\alpha)/g(\alpha)$.

Let $\mathcal{L}(lP_\infty)$ be the Riemann-Roch space over $\mathbb{F}_q$. Formally, this is

$$\mathcal{L}(lP_\infty) = \{h \in F \setminus \{0\} : \nu_{P_\infty}(h) \geq -l\} \cup \{0\},$$

where ν_{P_∞} is the discrete valuation of P_∞. Informally (in our running example), this should be thought of as the set of rational functions $f(X)/g(X)$ so that $\deg(g(X)) - \deg(f(X)) \geq -l$. In particular, the number of poles of f/g is at least the number of roots, minus l. It would be tempting, in this example, to think of these as degree $\leq l$ polynomials; all but at most l of the powers of X in the numerator are "canceled" in the denominator. Of course, there are many problems with this intuition, but it turns out that this indeed works out in some sense. In particular, it can be shown that the dimension of this space is at least $l - g + 1$. When $g = 0$ (as in our running example), it is exactly $l + 1$, the same as the dimension of the space of degree $\leq l$ polynomials.

More generally (whatever the genus), for any rational place P, we may write a function $h \in \mathcal{L}_m(lP_\infty)$ as

$$h = \sum_{j=0}^{\infty} h_j T^j, \tag{4}$$

where T is a local parameter of P, and it turns out that h is uniquely determined by the first $l+1$ coefficients $h_0, h_1, \ldots, h_{l+1}$.

Now let F_m be the constant extension $\mathbb{F}_{q^m} \cdot F$, and let $\mathcal{L}_m(lP_\infty)$ be the corresponding Riemann-Roch space. This has the same dimension over $\mathbb{F}_{q^m}$ as $\mathcal{L}(lP_\infty)$ does over $\mathbb{F}_q$. Now we consider the algebraic geometry code defined by

$$C(m; l) := \{(h(P_1), \ldots, h(P_n)) \,:\, h \in \mathcal{L}_m(lP_\infty)\}.$$

Following the intuition that $h(P_i)$ denotes function evaluation, this definition looks syntactically the same as a standard polynomial evaluation code, and should be thought of that way. This is an $\mathbb{F}_{q^m}$-linear code over $\mathbb{F}_{q^m}$, with block length n and dimension at least $l - g + 1$.

Step 1. List decoding with structured lists to list recovery with structured lists. With the preliminaries (and some basic, if possibly misleading, intuition for the reader unfamiliar with AG codes) out of the way, we press on with the argument.

Fix a parameter k, and consider a general AG code $C(m; k+2g-1)$, with the notation above. (We will fix a particular AG (sub)code later, by choosing a function field and by choosing a subcode). Let $S_1, \ldots, S_n \subset \mathbb{F}_{q^m}$ be lists of size at most ℓ corresponding to each coordinate. We first show that $C(m; k+2g-1)$ is $(1-\beta, \ell, L)$-list recoverable for some β to be chosen below, where the list size is very large, but the list is structured. In [GX13], the approach (similar to that in [GW13]) is as follows.

1. We will first find a low degree interpolating linear polynomial (whose coefficients live in Riemann-Roch spaces)

$$Q(Y_1, \ldots, Y_s) = A_0 + A_1 Y + \cdots + A_s Y_s$$

so that $A_i \in \mathcal{L}_m(DP_\infty)$ and $A_0 \in \mathcal{L}_m((D+k+2g-1)P_\infty)$, for some parameter k to be chosen later, for

$$D = \left\lfloor \frac{\ell n - k + (s-1)g + 1}{s+1} \right\rfloor,$$

and subject to ℓn linear constraints over $\mathbb{F}_{q^m}$. Before we list the constraints, notice that the number of degrees of freedom in Q is

$$s(D-g+1) + D + k + g,$$

because the $\mathbb{F}_{q^m}$-dimension of $\mathcal{L}_m((D+k+2g-1)P_\infty)$ is at least $D+k+g$, and the $\mathbb{F}_{q^m}$-dimension of $\mathcal{L}_m((DP_\infty))$ is at least $D-g+1$. Thus, the choice of D shows that the dimension of this space of interpolating polynomials is greater than ℓn. Thus, we will be able to find such a Q that satisfies the ℓn following ℓn constraints. For each $i \in [n]$ and for all $y \in S_i$, we have the constraint that

$$A_0(P_i) + A_1(P_i) y + A_2(P_i) y^q + \cdots + A_s(P_i) y^{q^{s-1}} = 0.$$

2. With this polynomial Q in hand, we observe that if $h \in \mathcal{L}_m((k+2g-1)P_\infty)$ whose encoding has $h(P_i) \in S_i$ for at least βn positions i, for $\beta n > D+k+2g-1$, then $Q(h, h^\sigma, \ldots, h^{\sigma^{s-1}}) = 0$, where h^σ denotes the extension of the Frobenius automorphism $\alpha \mapsto \alpha^q$ on $\mathbb{F}_{q^m}$ to $\mathcal{L}_m(lP_\infty)$. This proof (Lemma 4.7 in [GX12b]) remains unchanged when we pass to list recovery from list decoding. Briefly, this agreement means that

$$Q(h, \ldots, h^{\sigma^{s-1}})(P_i) = A_0(P_i) + A_1(P_i) h(P_i) + \cdots + A_s(P_i) h(P_i)^{q^{s-1}} = 0$$

for at least βn values of i, and so the function $Q(h, h^\sigma, \ldots, h^{\sigma^{s-1}})$ (which lies in $\mathcal{L}_m((D+k+2g-1)P_\infty)$; as per the intuition above, we are thinking of these as roughly analogous to degree-$(D+k+2g-1)$ polynomials) has at least $\beta n \geq D+k+2g-1$ roots, and hence is the zero function.

3. Thus, any element $h \in \mathcal{L}_m((k+2g-1)P_\infty)$ that agrees with at least βn lists also satisfies $Q(h, \ldots, h^{\sigma^{s-1}}) = 0$. It remains to analyze the space of these solutions, and to show that they are nicely structured. This requires one more step, which goes through without change. More precisely, [GX13] takes a subcode of $C(m; k+2g-1)$; this subcode will still have a large list size, but the list will be structured. This resulting code, denoted $C(m; k+2g-1 | \mathbb{F}_{q^m}^k)$, has dimension k. (Recall that $C(m; k+2g-1)$ has dimension $k+g$, so we have reduced the

dimension by g.) We refer the reader to [GX13] for the details, as they do not matter for us. At the end of the day, the analysis of [GX13] (Lemma 4.8 in the full version [GX12b]) applies unchanged to show that the set of messages h in this new code that are solutions to this equation lie in a structured space: more precisely, the coefficients $(h_0, h_1, \ldots, h_{k+2g-1})$ as in (4) belong to an $(s-1, m)$-*ultra periodic subspace* of $\mathbb{F}_q^{m(k+2g-1)}$. For us, the precise definition of this does not matter, as we may use the rest of [GX13] as a black box.

4. Before we move on, we summarize parameters. We have so far established that there is a code $C(m; k+2g-1|\mathbb{F}_{q^m}^k)$ that is list recoverable up to disagreement $1-\beta$ and with inner list sizes ℓ, resulting in a structured list. The requirement on β is:

$$\begin{aligned}\beta n &> D + k + 2g - 1 \\ &= \left\lfloor \frac{\ell n - k + (s-1)g + 1}{s+1} \right\rfloor + k + 2g - 1,\end{aligned}$$

and so it suffices to take

$$\begin{aligned}\beta n &> \frac{\ell n - k + (s-1)g + 1}{s+1} + k + 2g - 1 \\ &= \frac{1}{s+1}\left(\ell n + s(k-1) + g(3s+1)\right).\end{aligned}$$

Again, the dimension of the code is k and the length is n. It is $\mathbb{F}_{q^m}$-linear over $\mathbb{F}_{q^m}$.

Step 2. Taking a subcode. For this step, we may follow the argument of [GX13] without change. Briefly, to instantiate the AG code we use a function field from a *Garcia-Stichtenoth tower*. The parameters of this are as follows: we choose a prime power r, and let $q = r^2$. Then we choose an integer $e > 0$. There is a function field $F = K_e$ so that K_e has at least $n = r^{e-1}(r^2 - r) + 1$ rational places, and genus g_e bounded by r^e. This is the function field we will use. We remark that [GX13] has to do a bit of work here to show that one can actually find a description of the structured list efficiently, but it can be done. We plug in parameters to obtain the following Lemma, which is analogous to Theorem 4.14 in [GX12b].

Lemma A.2. *Let q be the even power of a prime, and choose $\ell, \varepsilon > 0$. There is a parameter $s = 1\ell/\varepsilon)$ so that the following holds. Let $m \geq s$ and let $R \in (0,1)$. Suppose that $\beta \geq R + \varepsilon + 3/\sqrt{q}$. Then for infinitely many n (all integers of the form $n = q^{e/2}(\sqrt{q}-1)$), there is a deterministic polynomial time construction of an $\mathbb{F}_{q^m}$-linear code C of block length n, dimension $k = Rn$, so that the following holds: for any sets $S_1, \ldots, S_n \subseteq \mathbb{F}_{q^m}$ with $|S_i| \leq \ell$ for all i, the set of messages leading to codewords $c \in C$ so that $c_i \in S_i$ for at least βn coordinates i is contained in one of $q^{O(mn)}$ possible $(s-1, m)$-ultra periodic $\mathbb{F}_q$-affine subspaces of $\mathbb{F}_q^{mk}$. Further, this collection of subspaces can be described in time* $\mathrm{poly}(n, m)$.

Proof. Our condition on β is that it is at least

$$\begin{aligned}\frac{\ell n + s(k-1) + g_e(3s+1)}{n(s+1)} &\leq \frac{\ell n + s(k-1) + n(3s+1)/(r-1)}{n(s+1)} \qquad \text{Using } g_e \leq n/(r-1) \\ &= \frac{\ell + s(R - 1/n) + (3s+1)/(r-1)}{s+1}.\end{aligned}$$

Choosing $s = O(\ell/\varepsilon)$ and using the fact that $r = \sqrt{q}$ gives the conclusion. □

With this lemma in hand, we may proceed exactly as the proof in [GX13]; indeed, it is exactly the same code, and we exactly the same conclusion on the structure of the candidate messages. The basic idea is to choose a subset of messages carefully via a *cascaded subspace design.* This ensures that the number of legitimate messages remaining in the list is small, and further that they can be found efficiently.

We briefly go through parameters, again referring the reader to the discussion in [GX13, GK16b] for details. We will fix

$$s = O(\ell/\varepsilon), \qquad \text{and} \qquad m = O\left(\frac{\ell}{\varepsilon^2} \cdot \log_q(\ell/\varepsilon)\right). \tag{5}$$

We now trace these choices through the analysis of [GX13, GX14].

Remark A.3. The reader familiar with these sorts of arguments might expect us to set $m = \ell/\varepsilon^2$, and indeed this would be sufficient if we could allow q to be sufficiently large. However, in this case, setting m this way would result in a requirement that $q \geq \ell/\varepsilon^2$. We would like q to be independent of ℓ for the next concatenation step to work (of course, the alphabet size q^m must be larger than ℓ), and this requires us to take m slightly larger. This loss comes out in the final list size.

Without defining a cascaded subspace design, we will just mention that it is a sequence of T subspace designs; a cascaded subspace design comes with vectors of parameters $(r_0, \ldots, r_T)$, $(m_0, \ldots, m_T)$, and $(d_0, \ldots, d_{T_1})$. For $i = 1, \ldots, T$, the i'th subspace design in this sequence is a (r_{i-1}, r_i)-strong subspace design in $\mathbb{F}_q^{m_{i-1}}$, of cardinality m_i/m_{i-1}, and dimension d_{i-1}. For our argument all that matters is that we may find explicit cascaded subspace designs:

Theorem A.4 (Follows from Theorem 6 in [GK16b])**.** *For all $\zeta \in (0,1)$ and for all r, m with $r \leq \zeta m/4$, and for all prime powers q so that $2r/\zeta < q^{\zeta m/(2r)}$, there exists an explicit collection of $M \geq q^{\Omega(\zeta m/r)}/(2r)$ subspaces in $\mathbb{F}_q^m$, each of codimension at most ζm, which form a $(r, r^2/\zeta)$-strong subspace design.*

Remark A.5. In [GK16b], the theorem is stated for $(r, r/\zeta)$-weak subspace designs; however, as is noted in that work, a (A, B)-weak subspace design is also a (A, AB)-strong subspace design, which yields our version of the theorem.

Below, we will use Theorem A.4 in order to instantiate a cascaded subspace design. The reason we want to do this is because of Lemma 5.6 in [GX12b]:

Lemma A.6 (Lemma 5.6 in [GX12b])**.** *Let $\mathcal{M}$ be a $(r_0, r_1, \ldots, r_T)$-cascaded subspace design with length vector $(m_0, m_1, \ldots, m_T)$. Let A be an (r, m)-ultra periodic affine subspace of $\mathbb{F}_q^{m_T}$. Then the dimension of the affine space $A \cap U(\mathcal{M})$ is at most r_T, where $U(\mathcal{M})$ denotes the* canonical subspace *of $\mathcal{M}$.*

We have not defined a canonical subspace, and we refer the reader to [GX12b] for details; the important thing for us is that we wish to construct a cascaded subspace design $\mathcal{M}$ so that r_T is small, m_T is equal to mk, and so that $r_0 = s - 1$ and $m_0 = m$. This will allow us to choose a subcode of the code from Lemma A.2 by restricting the space of messages to the canonical subspace $U(\mathcal{M})$, and this will be the $\mathbb{F}_q$-linear code (over $\mathbb{F}_q^m$) that we are after.

We may use Theorem A.4 to instantiate such a cascaded subspace design as follows (the derivation below follows the proof of Lemma 5.7 in [GX12b]). We choose $\zeta_i = \varepsilon/2^i$, $r_0 = s-1$, and $r_i = r_{i-1}^2/\zeta_i$.

We choose $m_0 = m$ and we will define $m_i = m_{i-1} \cdot q^{\sqrt{m_{i-1}}}$. We will continue up to $i = T$, choosing T so that $m_T = mk$. At this point, we must deal with the detail that there may be no such T; to deal with this we do exactly as in the proof of Lemma 5.7 in [GX12b] and modify our last two choices of m_{T-1}, m_T so that $m_T \leq mk$ but is close (within an additive $\log_q^2(km)$); for our final subspace, we will pad the m_T-dimensional vectors with 0's in order to form a subspace in $\mathbb{F}_q^{mk}$ with the same dimension. Choosing $m_T \approx mk$ puts $T = O(\log^*(mk))$, and an argument by induction shows that

$$r_T \leq \frac{s^{2^T} 2^{4^T}}{\varepsilon^{2^T - 1}},$$

which with this choice of T implies that

$$r_T = \left(\frac{\ell}{\varepsilon}\right)^{2^{O(\log^*(mk))}}.$$

With these choices, we instantiate T subspace designs via Theorem A.4, with $m \leftarrow m_i, r \leftarrow r_i$, and $\zeta \leftarrow \zeta_i$. We check that the requirements of Theorem A.4 are satisfied, beginning with the requirement that $r_i \leq \zeta_i m_i / 4$. Since $m_i \zeta_i$ grows much faster than r_i as i increases, it suffices to check this for $i = 0$, when we require $r_0 \leq \zeta_0 m_0$, or $s - 1 \leq m\varepsilon/8$. Our choices of m and s in (5) satisfy this.

The next requirement is that $2r_i/\zeta_i \leq q^{\zeta_i m_i/(2r_i)}$ for all i. Again, the right hand side grows much faster than the left, and so we establish this for $i = 0$, requiring that

$$\frac{4(s-1)}{\varepsilon} \leq q^{\varepsilon m/4(s-1)}.$$

With our choices of m and s, this requirement is that

$$\frac{\ell}{\varepsilon^2} \leq q^{O(\log_q(\ell/\varepsilon))},$$

which is true.

Thus, Theorem A.4 provides us with a cascaded subspace design with the given parameters. As mentioned above, we may then use Lemma A.6 to choose an appropriate subcode of our AG code from Lemma A.2. We have chosen the parameters above so that $(r_0, m_0) = (s-1, m)$, precisely the guarantee of Lemma A.2. Thus, the final bound on the dimension of the intersection with any affine ultra-periodic subspace (that is, with any space of potential messages) is $r_T \leq (\ell/\varepsilon)^{2^{O(\log^*(mk))}}$, which gives a final bound on the dimension of the output list. Finally, we observe (as in Observation 5.5 of [GX12b]) that the dimension of the resulting subcode is at least $(1 - \sum_i \zeta_i) m_T = (1 - \varepsilon)mk$. Thus the final code has dimension at least $(1 - \varepsilon)mk$ over $\mathbb{F}_q^{km}$, and hence the final rate is at least $(1 - \varepsilon)R$. Observing that q must be at least ε^{-2} for the $1/\sqrt{q}$ term in Lemma A.2 to be absorbed into the additive ε factor, we arrive at the following theorem.

Theorem A.7. *Let q be an even power of a prime, and choose $\ell, \varepsilon > 0$, so that $q \geq \varepsilon^{-2}$. Choose $\rho \in (0,1)$. There is an $m_{min} = O(\ell \log_q(\ell/\varepsilon)/\varepsilon^2)$ so that the following holds for all $m \geq m_{min}$. For infinitely many n (all n of the form $q^{e/2}(\sqrt{q} - 1)$ for any integer e), there is a deterministic polynomial time construction of an $\mathbb{F}_q$-linear code $C : \mathbb{F}_{q^m}^{\rho n} \to \mathbb{F}_{q^m}^n$ of rate ρ and relative distance $1 - \rho - O(\varepsilon)$ that is $(1 - \rho - \varepsilon, \ell, L)$-list recoverable in time $\mathrm{poly}(n, L)$, returning a list that is contained in a subspace over $\mathbb{F}_q$ of dimension at most*

$$\left(\frac{\ell}{\varepsilon}\right)^{2^{O(\log^*(mk))}}.$$

We note that the distance of the code comes from the fact that it is a subcode of $C(m; k+3g_e-1)$, which has distance at least $n-(k+2g_e-1) = n-2g_e-k+1$. In the above parameter regime, the genus g_e satisfies $g_e \le n/(r-1) = n/(\sqrt{q}-1) = O(\varepsilon n)$. Thus, the relative distance of the final code is at least $(n-2g_e-k+1)/n \ge 1-O(\varepsilon)-\rho$.

Step 3. Concatenating to obtain $\mathbb{F}_q$-linear codes over $\mathbb{F}_q$. Theorem A.7 gives codes over $\mathbb{F}_{q^m}$ which are $\mathbb{F}_q$-linear. For our purposes, to prove Theorem A.1, we require codes over $\mathbb{F}_q$ which are $\mathbb{F}_q$-linear. Thus, we will concatenate these codes with random $\mathbb{F}_q$-linear codes from Corollary 2.2, and apply Lemma B.1 about the concatenation of list recoverable codes which we state and prove in Appendix B.

In more detail, we choose parameters as follows. Let $\varepsilon > 0$ and let $\varepsilon' = \varepsilon/2$, and choose any integer ℓ and any block length N. Fix a constant c and parameters m and e which will be determined below. Choose an even prime power q so that

$$q \ge \max\left\{\ell^{c/\varepsilon}, \varepsilon^{-c}\right\}.$$

Let C_{in} be a random q-ary linear code of rate $\rho_{in} = 1-\varepsilon'$ of length m/ρ_{in}. By Corollary 2.2, there exists an $\mathbb{F}_q$ linear code C_{in} with rate $\rho_{in} = 1-\varepsilon'$ and block length m/ρ_{in} which is $(\alpha_{in}, \ell_{in}, L_{in})$-list recoverable, for $\alpha_{in} = \varepsilon'/2$, $\ell_{in} = \ell$, and $L_{in} = q^{2c\ell/\varepsilon'}$. We note that we can choose c large enough to ensure that the hypothesis of Corollary 2.2 hold.

Let C_{out} be the codes from Theorem A.7, instantiated with rate $\rho_{in} = 1-\varepsilon'$, $\varepsilon \leftarrow \varepsilon'/2$ and $\ell \leftarrow L_{in}$. With these parameters, we will get a code over $\mathbb{F}_{q^m}$ of length $n = q^{e/2}(\sqrt{q}-1)$ which is $(\alpha_{out}, L_{in}, L_{out})$-list recoverable, where

$$\begin{aligned} L_{out} &= \exp_q\left((L_{in}/\varepsilon')^{2^{O(\log^*(mk))}}\right) \\ &= \exp_q\left(\left(\frac{q^{2c\ell/\varepsilon'}}{\varepsilon'}\right)^{2^{O(\log^*(mk))}}\right) \end{aligned}$$

and where

$$\alpha_{out} = 1-\rho_{in}-\varepsilon' = \varepsilon'/2.$$

Let $m_{\min}$ be as in Theorem A.7, so that

$$m_{\min} = O(L_{in}\log_q(L_{in}/\varepsilon')/(\varepsilon')^2) = O\left(\frac{q^{c\ell/\varepsilon'}c\ell}{(\varepsilon')^3}\right).$$

We will choose m so that

$$m_{\min} \le m \le q \cdot m_{\min}. \tag{6}$$

Notice that, given the definition of $m_{\min} = O(q^{c\ell/\varepsilon'}c\ell/(\varepsilon')^3)$, choosing m slightly larger than $m_{\min}$—as large as $q \cdot m_{\min}$—amounts to replacing the constant c with $c+1$. Thus, the choices of m and c (subject to (6)) will not affect the list recoverability of C_{out}, but they will affect the block length of the concatenated code.

Formally, Lemma B.1 implies that the concatenated code has rate $\rho_{in}\cdot\rho_{out} = (1-\varepsilon')^2 \ge 1-\varepsilon$, and is $(\alpha_{in}\alpha_{out}, \ell, L_{out})$-list recoverable. Here, we have

$$\alpha_{in}\alpha_{out} = (\varepsilon')^2/4 = \Omega(\varepsilon^2),$$

which is what is claimed in Theorem A.1. The output list size claimed in Theorem A.1 follows from the choice of m and our guarantee on L_{out}. We note that the concatenated code will have message length $K = mk$, and so we write $\log^*(mk) = \log^*(K)$.

Finally, we choose m and e. At this point, the choice of these parameters (subject to (6)) will not affect that list recoverability of the concatenated code, but they do control the block length of the code and the running time of the decoding algorithm. The block length is

$$\frac{m}{\rho_{in}} \cdot q^{e/2}(\sqrt{q} - 1).$$

In order to prove that we can come up with such codes for all sufficiently large block lengths N, as required in the statement of Theorem A.1, we must show that for all sufficiently large N, we can choose m satisfying (6) and e so that

$$N = \frac{m}{\rho_{in}} \cdot q^{e/2}(\sqrt{q} - 1).$$

That is, we want to find an integer e so that

$$\frac{N \cdot (1 - \varepsilon/2)}{q^{e/2}(\sqrt{q} - 1)} \in [m_{\min}, q \cdot m_{\min}].$$

However, we have chosen this window for m to be large enough so that such an e exists as long as N is sufficiently large (in terms of q, ℓ, ε). More precisely, for some large enough constant C, we require

$$N \geq q^{C\ell/\varepsilon},$$

which is our choice of N_0 in Theorem A.1.

Now we verify the running time of the list recovery algorithm. The outer code C_{out} can be list recovered in time $\text{poly}(n, L_{out})$ by Theorem A.7. The base code can be list recovered by brute force in time $q^{O(m)} = \exp_q\left(O\left(q^{2(c+1)\ell/\varepsilon}\ell/\varepsilon^3\right)\right) = \text{poly}(L_{out})$. Lemma B.1 implies that the final running time is $\text{poly}(N, L)$, where $L = L_{out}$ is the final list size and N is the block length of the concatenated code.

B Concatenation of list recoverable codes

In this appendix, we prove the following lemma, which says that the concatenation of two list recoverable codes is again list recoverable.

Lemma B.1. *Let $C_{out} : \mathbb{F}_{q^s}^{\rho_{out} \cdot n} \to \mathbb{F}_{q^s}^{n}$ be an $(\alpha_{out}, \ell_{out}, L_{out})$-list recoverable code, with a list recovery algorithm running in time T_{out}. Let $C_{in} : \mathbb{F}_q^s \to \mathbb{F}_q^{s/\rho_{in}}$ be $(\alpha_{in}, \ell_{in}, L_{in})$-list recoverable for $L_{in} = \ell_{out}$, with a list recovery algorithm running in time T_{in}. Let $C : \mathbb{F}_q^{s \cdot \rho_{out} \cdot n} \to \mathbb{F}_q^{sn/\rho_{in}}$ be the code obtained from concatenating C_{out} with C_{in}: that is, each symbol of $c \in C_{out}$ is then encoded using C_{in}. Then C is $(\alpha_{out} \cdot \alpha_{in}, \ell_{in}, L_{out})$-list recoverable in time $T_{out} + O(n \cdot T_{in})$ and has rate $\rho_{in} \cdot \rho_{out}$.*

Proof. For $i \in [n]$ and $j \in [s/\rho_{in}]$, let $S_{i,j} \subseteq \mathbb{F}_q$ be a list of at most ℓ_{in} possible symbols for the coordinate $C(x)_{i,j} := C(x)_{(i-1)\cdot(s/\rho_{in})+j}$, which is the j-th coordinate of $C_{in}(C_{out}(x)_i)$.

Suppose that for at most a $\alpha_{out} \cdot \alpha_{in}$ fraction of coordinates (i,j), $C(x)_{i,j} \notin S_{i,j}$. Then by Markov's inequality, for at most an α_{out} fraction of $i \in [n]$, the blocks $C_{in}(C_{out}(x)_i)$ have more than α_{in} fraction of the $j \in [s/\rho_{in}]$ so that $C(x)_{i,j} \notin S_{i,j}$. Thus, we may list recover each block $C_{in}(C_{out}(x)_i)$ to obtain a list $S_i \subseteq \mathbb{F}_{q^s}$ of at most $L_{in} = \ell_{out}$ possible symbols for $C_{out}(x)_i$, and the above reasoning shows that $C_{out}(x)_i \notin S_i$ for at most $\alpha_{out} n$ values of i. Now we may run C_{out}'s list recovery algorithm to obtain a final list of size L_{out}.

Finally, the claim about the rate follows from the definition of concatenation. □

C Details for the proof of Lemma 4.1

In this appendix we work out the computations for the proof of Lemma 4.1.

As in the proof of that lemma, suppose by induction that we have applied Lemma 4.2 i times in order to obtain a code $C^{\otimes(i+1)}$ which is $(\varepsilon \cdot (\delta/d_0)^{t-1-i})$-approximately $(\alpha \cdot s_0^{-ti}, \ell, L^{s_0^{ti} \cdot \log^i(L)})$-locally list recoverable, with query complexity $n \cdot s_0^{ti} \log^i(L)$, in time $T_0(n) \cdot s_0^{ti} \log^i(L)$, and with pre-processing time $P_0(n) \cdot s_0^{ti} \log^i(L) + L^{s_0^{ti} \log^i(L)}$. Then we apply Lemma 4.2 with the following parameters. The approximately locally list recoverable code C in Lemma 4.2 is the code $C^{\otimes(i+1)}$ constructed so far. Thus, in the statement of Lemma 4.2 we take $\alpha \leftarrow \alpha \cdot s_0^{-ti}$ and $L \leftarrow L^{s_0^{ti} \cdot \log^i(L)}$. The globally list recoverable code C' in Lemma 4.2 is a copy of the globally list recoverable code C guaranteed by the lemma statement. Thus, in the statement of Lemma 4.2 we take $\alpha' \leftarrow \alpha$, $\delta' \leftarrow \delta$, and $L' \leftarrow L$. In the statement of Lemma 4.2 we will choose $s \leftarrow s'$ so that

$$s' = s_0 \left(\frac{\delta}{d_0}\right)^{-b_0(t-1-i)}.$$

Notice that s' does indeed satisfy the requirement of Lemma 4.2 as applied to $C^{\otimes(i+1)}$, by our assumption about $C^{\otimes(i+1)}$ and our choices of α', δ' above. Moreover, provided that ε is sufficiently small compared to d_0, we have

$$s' = s_0 \left(\left(\frac{d_0}{\delta}\right)^{b_0}\right)^{t-1-i} \leq \frac{1}{2} \cdot s_0 \left(\left(\frac{1}{\varepsilon\delta\alpha}\right)^{b_0}\right)^{t-1-i} \leq \frac{1}{2} s_0^t. \tag{7}$$

Now we apply Lemma 4.2 to get a code $C^{\otimes(i+2)}$ which is ε''-approximately (α'', ℓ, L'')-locally list recoverable with the following parameters.

- The approximability parameter ε'' is given by

$$\varepsilon'' = \varepsilon \left(\frac{\delta}{d_0}\right)^{t-1-i} \left(\frac{d_0}{\delta}\right) = \varepsilon \left(\frac{\delta}{d_0}\right)^{t-1-(i+1)}.$$

- The parameter α'' is given by

$$\begin{aligned} \alpha'' &= \alpha \cdot s_0^{-ti}/(s') \\ &\geq \alpha \cdot s_0^{-ti} s_0^{-t} \\ &= \alpha \cdot s_0^{-t(i+1)}, \end{aligned}$$

  using (7).

- The output list size L'' satisfies

$$\begin{aligned} L'' &= \left(L^{s_0^{ti}\cdot\log^i(L)}\right)^{s'\log(L)} \\ &\le L^{s_0^{ti}\cdot\log^i(L)\cdot s_0^t\log(L)} \\ &= L^{s_0^{t(i+1)}\cdot\log^{i+1}(L)} \end{aligned}$$

  again using (7)

- The query complexity is at most

$$\begin{aligned} n\cdot s_0^{ti}\log^i(L)\cdot s'\log(L) &\le s_0^{ti}\log^i(L)\cdot s_0^t\cdot\log(L) \\ &\le s_0^{t(i+1)}\log^{i+1}(L) \end{aligned}$$

  using (7) again. The running time obeys the same recurrence relation as the query complexity, and so the recovery algorithm for $C^{(i+1)}$ also runs in time $T_0(n)\cdot s_0^{t(i+1)}\log^i(L)$.

- Finally, the preprocessing time is at most

$$\begin{aligned} &\left(P_0(n)\cdot s_0^{ti}\log^i(L)+L^{s^{ti}\log^i(L)}\right)s'\log(L)+L^{s_0^{ti}\log^i(L)s'\log(L)} \\ &\le \left(P_0(n)\cdot s_0^{ti}\log^i(L)+L^{s^{ti}\log^i(L)}\right)s_0^t\log(L)+L^{s_0^{ti}\log^i(L)\frac{s_0^t}{2}\log(L)} \\ &\le P_0(n)\cdot s_0^{t(i+1)}\log^{i+1}(L)+L^{s^{ti}\log^i(L)}s_0^t\log(L)+L^{\frac{1}{2}s_0^{t(i+1)}\log^{i+1}(L)} \\ &\le P_0(n)\cdot s_0^{t(i+1)}\log^{i+1}(L)+L^{s_0^{t(i+1)}\log^{i+1}(L)}. \end{aligned}$$

  Above, we used (7), along with the assumption that $L>1$ and that s_0 is sufficiently large (which follows from our assumption that ε is sufficiently small).

Thus, we conclude that $C^{\otimes(i+2)}$ is $(\varepsilon\cdot(\delta/d_0)^{t-1-(i+1)})$-approximately $(\alpha\cdot s_0^{-t(i+1)},\ell,L^{s_0^{t(i+1)}\cdot\log^{i+1}(L)})$-locally list recoverable, with query complexity at most $n\cdot s_0^{t(i+1)}\log^i(L)$ in time $T_0(n)\cdot s_0^{t(i+1)}\log^i(L)$, and with preprocessing time at most $P_0(n)\cdot s_0^{t(i+1)}\log^{i+1}(L)+L^{s_0^{t(i+1)}\log^{i+1}(L)}$, which establishes the inductive hypothesis for $i+1$.

By induction, we conclude that $C^{\otimes t}$ is ε-approximately $(\alpha\cdot s_0^{-t^2},\ell,L^{s_0^{t^2}\cdot\log^t(L)})$-locally list-recoverable with query complexity $n\cdot s_0^{t^2}\log^t(L)$ and in time $T_0(n)\cdot s_0^{t^2}\log^t(L)$. In particular, for any $s\ge s_0$, the conclusion of Lemma 4.1 holds.

D Useful concentration inequalities

We make use of the following two concentration inequalities. The first is the standard Hoeffding bound for independent 0/1-valued random variables.

Theorem D.1 (Hoeffding's inequality)**.** *Let $X_1,\dots,X_m\in[0,1]$ be independent random variables with mean μ. Then*

$$\mathbb{P}\left\{\left|\frac{1}{m}\sum_{i=1}^m X_i-\mu\right|\ge\gamma\right\}\le 2\exp(-2\gamma^2 m).$$

The second is a version of this inequality for random variables chosen without replacement. We use a version found in [GGR11].

Theorem D.2 ([GGR11], Lemma 5.1)**.** *Let $z_1, \ldots, z_n \in [0,1]$, and suppose that $S \subseteq [n]$ is a uniformly random set of size m. Then*

$$\mathbb{P}\left\{\left|\frac{1}{m}\sum_{i\in S} z_i - \frac{1}{n}\sum_{i=1}^{n} z_i\right| \geq \gamma\right\} \leq 2\exp(-2\gamma^2 m).$$

ECCC ISSN 1433-8092

https://eccc.weizmann.ac.il