9 New, Expander-Based List Decodable Codes

9.1 Introduction

In the previous chapters, we have already seen constructions of asymptotically good codes of good rate over both large alphabets (the AG-codes from Chapter 6) and the binary alphabet (the concatenated codes from Chapter 8), that are efficiently list decodable up to a "maximum" possible radius. By "maximum" possible radius we mean list decoding up to a fraction $(1 - 1/q - \varepsilon)$ of errors for q-ary codes. This translates into a fraction $(1 - \varepsilon)$ of errors for codes over large enough alphabets, and a fraction $(1/2 - \varepsilon)$ of errors for binary codes. For codes with such large list decodability, which we called "highly list decodable codes", the goal is to find efficient constructions that achieve good rate (typically of the form $\Omega(\varepsilon^a)$ for some reasonably small a), together with efficient list decoding algorithms.

The earlier results achieve fairly non-trivial trade-offs in this regard. The list decoding algorithm for AG-codes from Chapter 6 implies highly list decodable codes over an alphabet of size $O(1/\varepsilon^4)$ that have rate $\Omega(\varepsilon^2)$. The results of the previous chapter on concatenated codes give constructions of highly list decodable binary codes of rate $\Omega(\varepsilon^4)$.

One shortcoming of the former result is that the necessary AG-codes are very complicated to construct and the known decoding algorithms need a non-standard representation of the code for the claim of polynomial runtime to hold. Families of Reed-Solomon codes also offer similar list decodability with a rate of $\Omega(\varepsilon^2)$, but their alphabet size is at least as large as the blocklength and hence they do not achieve a alphabet size that is a constant dependent only on ε. In fact, other than AG-codes, there were no other known families of codes that are list decodable to a fraction $(1 - \varepsilon)$ of errors, have reasonably large rate, and are defined over a constant-sized alphabet.

The other shortcomings of the above mentioned results are that there is potential for improvement in the rate. The existential results (Chapter 5) show that a rate $\Omega(\varepsilon)$ is possible for highly list decodable codes over large alphabets, and a rate $\Omega(\varepsilon^2)$ is possible for binary codes. Thus the constructive results are not optimal with respect to the rate (though they are not off by very much).

In this chapter, we present novel constructions of list decodable codes that address the above shortcomings. Our codes are simple to construct and

V. Guruswami: List Decoding of Error-Correcting Codes, LNCS 3282, pp. 209-250, 2004.

decode, and share the common thread of using expander-like bipartite graphs as a component (the specific graphs that we use are referred to as *dispersers* in the literature). The bipartite graphs redistribute the symbols in such a way that information from a small fraction (say, ε) of correct nodes on the right is "dispersed" to a large fraction (say, $1/2$) of nodes on the left. They thereby enable the design of efficient decoding algorithms that correct a large number of errors through various forms of "voting" procedures.

The basic idea behind the constructions of this chapter will also find use later in Chapter 11, where we will present codes of good (in fact, near-optimal) rate that are uniquely decodable up to a large fraction of errors in *linear* time. This indicates the quite general applicability and power of the techniques used in this chapter.

An important combinatorial tool used in our constructions are "pseudolinear codes". We view the construction and use of pseudolinear codes as being of independent interest, and hope that it will find several applications in the future.[1] Pseudolinear codes possess the useful properties of efficient encoding and succinct representation which all linear codes automatically have, but they have the additional nice property that random pseudolinear codes (with suitable parameters) inherit the same list-of-L decoding properties as completely general random codes.

We next present a detailed statement of the results of this chapter, followed by an overview of the main techniques used.

9.2 Overview of Results and Techniques

9.2.1 Main Results

Our constructions of highly list decodable codes give the following:

(1) Codes of rate $\Omega(\varepsilon^2)$ over an alphabet of size $2^{O(\varepsilon^{-1}\log(1/\varepsilon))}$, list decodable up to a fraction $(1-\varepsilon)$ of errors in near-quadratic time.

(2a) Codes of rate $\Omega(\varepsilon)$ over an alphabet of size $2^{O(\varepsilon^{-1}\log(1/\varepsilon))}$, list decodable up to a fraction $(1-\varepsilon)$ of errors in sub-exponential time.

(2b) Binary codes of rate $\Omega(\varepsilon^3)$ list decodable up to a fraction $(1/2-\varepsilon)$ of errors in sub-exponential time.

(3) Codes of rate $\Omega(t^{-3}\varepsilon^{2+2/t})$ over an alphabet of size $O(1/\varepsilon^b)$, list decodable up to a fraction $(1-\varepsilon)$ of errors. Here $t \geq 1$ is an arbitrary integer and $b > t$ an arbitrary real.

The first three constructions $(1, 2a, 2b)$ use the expander-based approach mentioned in the introduction. The last construction does not use expanders/dispersers and is based on multiple concatenated codes combined

[1] Pseudolinear codes are also used in the next chapter on list decoding from erasures.

together by juxtaposing symbols together — we call such codes *juxtaposed codes*.[2] We discuss these codes also in this chapter since their construction has much the same motivation as that of (1). Moreover, they also use some of same machinery that construction (1) uses; specifically they too use pseudolinear codes as inner codes in a concatenated scheme. The main advantage of the juxtaposed code construction is that they can achieve better alphabet size than the construction (1), at the expense of a slight worsening of the rate.

The detailed specification of all parameters of our constructions are listed in Figure 9.1. We next present a discussion of the individual results and compare them with previously known constructions.

No	Alphabet	Decoding radius	Rate	Encoding time	Decoding time	Const. time (probabilistic)*
1	$2^{\varepsilon^{-1}\log(1/\varepsilon)}$	$1-\varepsilon$	ε^2	$n \log n$	$n^2 \log n$	$\log^2 n/\varepsilon$
2a	$2^{\varepsilon^{-1}\log(1/\varepsilon)}$	$1-\varepsilon$	ε	$n^{2(1-\gamma)} \log^2 n$	$2^{n^\gamma \log(1/\varepsilon)}$	$n^{2(1-\gamma)}/\varepsilon$
2b	2	$1/2-\varepsilon$	ε^3	$n^{2(1-\gamma)} \log^2 n$	$2^{n^\gamma \log(1/\varepsilon)}$	$n^{2(1-\gamma)}/\varepsilon$
3	$2^{\log^2(1/\varepsilon)}$	$1-\varepsilon$	$\varepsilon^2 \log^{-3}(1/\varepsilon)$	$n \log^2 n$	$n^{1/\varepsilon}$	$\log^2 n/\varepsilon^2$

Fig. 9.1. The parameters of our codes. n stands for the length of the code. For readability, the $O(\cdot)$ and $\Omega(\cdot)$ notation, and certain $\log^{O(1)}(1/\varepsilon)$ factors have been omitted. The value of γ is in the interval $(0, 1]$; its value influences the rate by a constant factor. The decoding radius shows the fraction of errors which the decoding algorithms can correct. .

$*$A detailed discussion on the construction times is presented later in this Section.

Our first code (1) enables efficient list decodability from up to a fraction $(1-\varepsilon)$ of errors, for an arbitrary constant $\varepsilon > 0$. Its distinguishing feature is the near-quadratic decoding time and fairly high (namely $\Omega(\varepsilon^2)$) rate, while maintaining a constant alphabet size. The only other known constructible codes with comparable parameters are certain families of algebraic-geometric codes [190, 65]. As discussed in Chapter 6 (specifically in Theorem 6.45), such AG-codes can achieve $\Omega(\varepsilon^2)$ rate and $O(1/\varepsilon^4)$ alphabet size. While they yield a much better alphabet size, AG-codes suffer from the drawback of complicated construction and decoding algorithms. It is only known how to list decode them in polynomial time using certain auxiliary advice (of polynomial size), and it not known how to compute this information in sub-exponential (randomized or deterministic) time (the reader might recall the

[2] Juxtaposed codes will be used again in the next chapter to obtain constructions of good codes with very high list decodability from erasures. Codes similar to our juxtaposition based constructions are also called multilevel concatenated codes in the literature [46], but we believe the term juxtaposed codes is more natural and we use this terminology.

discussion about this in Chapter 6, Section 6.3.9). Even regarding construction complexity, only very recently [167] showed how to construct the generator matrix of the necessary AG-codes in near-cubic time. In comparison, our construction time, although probabilistic, is essentially negligible.

The second code (2a) also enables list decodability up to a fraction $(1 - \varepsilon)$ of errors. Its distinguishing feature is the *optimal* $\Omega(\varepsilon)$ rate. The only previously known codes with such rate were purely random codes (even Reed-Solomon codes that have super-constant alphabet size only guarantee $\Omega(\varepsilon^2)$ rate). However, the best known decoding time for random codes is $2^{O(n)}$, and it is likely that no significantly better algorithm exists. Our codes also have significant random components; however, they can be decoded substantially faster in sub-exponential time. The binary version (2b) of the aforementioned codes, which correct up to a fraction $(1/2 - \varepsilon)$ of errors, also beat the $\Omega(\varepsilon^4)$ rate of best constructive codes from the previous chapter (specifically, the result of Theorem 8.11). They are only off by a factor of $O(\varepsilon)$ from the optimal $\Omega(\varepsilon^2)$ rate implied by the existential results of Chapter 5.

For the codes (3), the rate is not as good as the construction (1), but one can get substantial improvements in alphabet size for a relatively small worsening of the rate. For example, as listed in Figure 9.1, it can achieve a rate of $\Omega(\varepsilon^2 \log^{-3}(1/\varepsilon))$ for an alphabet size of $2^{O(\log^2(1/\varepsilon))}$. By worsening the rate further, it is even possible to achieve an alphabet size better than $O(1/\varepsilon^4)$, which is the alphabet size of the best known AG-codes that are list decodable up to a fraction $(1 - \varepsilon)$ of errors (see Theorem 9.25).

Construction times. All of our constructions use the probabilistic method to obtain certain "gadgets" which are then used together with explicitly specified objects. The probabilistic method generates such building blocks with high probability. Therefore, our probabilistic construction algorithms are *randomized Monte Carlo*, and the claimed list decodability property holds with high probability over the choice of the random components. We note, however, that our probabilistic algorithms using R random bits can be derandomized and converted into deterministic algorithms using $O(R)$ space and running in $2^{O(R)}$ time in a straightforward manner. (The resulting code will be *guaranteed* to have the claimed list decodability property, i.e., the derandomization includes "verification" as well.) For the codes (1), a naive derandomization would only give a quasi-polynomial time construction. Nevertheless, by using the method of conditional expectations for derandomization, we will show the code can be constructed deterministically in time $n^{O(\varepsilon^{-1}\log(1/\varepsilon))}$. Similarly, for our constructions (2a,2b), a conditional expectations based derandomization enables a deterministic construction in (roughly) $2^{O(n^{1-\gamma})}$ time. Note that the both the probabilistic and deterministic construction times of (2a,2b) get worse as the decoding time gets better and better.

We stress that modulo the gadget construction, generating each symbol of a codeword can be done in *polylogarithmic time*.

9.2.2 Our Techniques

Expander-Based Constructions At a high level, the codes (1), (2a,2b) are all constructed using a similar scheme. The basic components of the constructions are: a "left" code (say, C) and a "dispersing" bipartite graph G, and in the case of binary codes, a "right" binary code C'. The left code C is typically a concatenation of efficient list decodable codes, namely Reed-Solomon codes and certain good list decodable "pseudolinear" codes whose existence we prove in Section 9.3. Such pseudolinear codes can either be found by brute-force or, one can pick a code at random and thus get a much faster probabilistic construction that works with high probability. The bipartite graph G has a weak form of expansion property, namely that the neighborhood of every reasonable sized subset of the left side (say, consisting of a fraction $1/2$ of the left nodes) misses at most a fraction ε of the nodes on the right. The technical term in the literature for such a graph is usually *disperser*, though we find it convenient to use the umbrella term expander to loosely refer to such graphs in the sequel.[3]

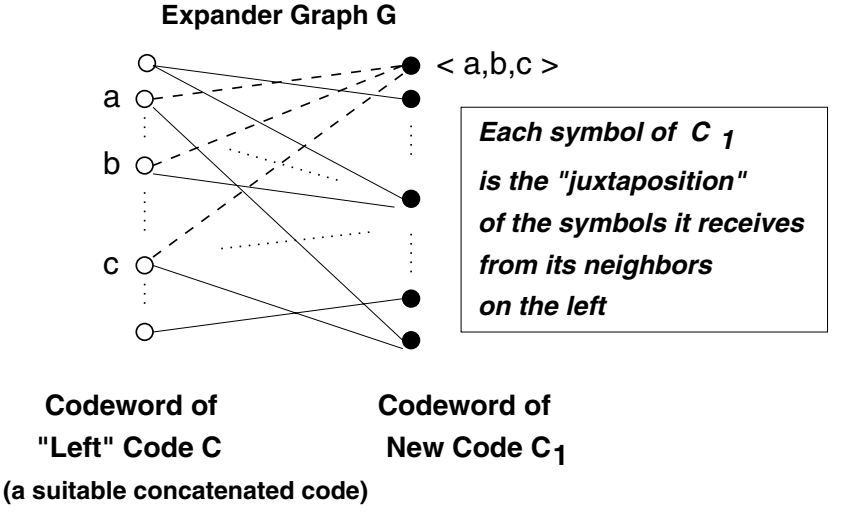

Fig. 9.2. Basic structure of our code constructions. To get binary code constructions, each symbol of C_1 is further concatenated with a good, constant-sized binary code.

[3] In Chapter 11, where we will also make use of expanders, we will use stronger "pseudorandom" properties of expander graphs, and not just their dispersion property.

Given the above components, the codes are constructed as follows. For each codeword x of C, we construct a new codeword y by distributing the symbols of x from left to right according to the edges in G. The juxtaposition of symbols "sent" to each right node of G forms a symbol of the codeword y of the final code C_1. The code C_1 will thus be defined over a large alphabet. See Figure 9.2 for a sketch of the basic construction scheme. For construction (2b), in order to get a binary code, we add a final level of concatenation with an appropriate binary code C'. This is similar to the construction due to Alon et al in [6]. Our contribution is in the design of efficient decoding algorithms to correct a large fraction of errors for such code constructions.

The role of the dispersing graph G is, roughly speaking, to convert an arbitrary distribution of errors that could exist between the various blocks of the (concatenated) left code C into a near-uniform distribution. This permits recovery of a (somewhat corrupted) received word x for C from a heavily corrupted received word y for the code C_1, using a certain "voting" scheme. The voting scheme we use is very simple: each position of y votes for all positions of x which are connected to it by an edge of G. This allows us to collect a list of potential symbols for each position of x. These lists are then used by a suitable decoding algorithm for C to finish the decoding.

The specifics of the implementation of the above ideas depend on the actual code construction. For the code (1), we take the left code C to be a concatenation of a Reed-Solomon code and a suitable pseudolinear code. Such a code can be list decoded in near-quadratic time using the Reed-Solomon decoding algorithms discussed in Chapter 6. The codes (2a,2b) are constructed by picking C to be a concatenation of a constant number of levels of "pseudolinear" codes with an outermost Reed-Solomon code (we call such codes *multi-concatenated codes*). The pseudolinear codes can perform list decoding when given as input a vector of lists, one per codeword position, such that at least half of the lists contain the correct symbol. The important fact is that such pseudolinear codes exist with a fixed constant rate that is *independent* of the length of the lists that are involved. This allows the decoding algorithm to propagate the candidate symbols through the concatenation levels while decreasing the rate only by a small factor at each level. The parameters are so picked that the decoding of each of these pseudolinear codes as well as the overall code can be done in sub-exponential time.

Juxtaposed Code Constructions The second approach behind our code constructions, which is used in Section 9.6, is aimed at obtaining similar (or slightly worse) rates using smaller alphabet size, and is the basis of the constructions described in Section 9.6. In this approach, multiple Reed-Solomon codes (of varying rates) are concatenated with *several* different inner codes (of varying rate and list decodability). Corresponding to each Reed-Solomon and inner code pair, we get one concatenated codeword, and the final encoding of a message is obtained by "juxtaposing together" the symbols from the various individual concatenated codewords.

The purpose of using multiple concatenated codes is that depending on the distribution of errors in the received word, the portions of it corresponding to a significant fraction of a *certain* inner encoding (that depends on the level of non-uniformity in the distribution of errors) will have relatively few errors. These can then be decoded to provide useful information about a large fraction of symbols to the decoder of the *corresponding* outer Reed-Solomon code. Essentially, depending on how (non)-uniformly the errors are distributed, a certain concatenated code "kicks in" and enables recovery of the message. A conceptually similar idea was used by Albanese et al in their work on Priority encoding transmission (PET) [3].

The use of multiple concatenated codes reduces the rate compared to the expander-based constructions, but we gain in the alphabet size. For example, for a near-quadratic (namely, $\Omega(\varepsilon^2 \log^{-O(1)}(1/\varepsilon))$) rate, the alphabet size can be quasi-polynomial as opposed to exponential in $1/\varepsilon$.

9.2.3 A Useful Definition

For our results, the following (more general) notion of good list decodability proves extremely useful — for purposes of disambiguation from (e, ℓ)-list decodability, we call this notion "list recoverability".

Definition 9.1. *For α, $0 < \alpha < 1$, and integers $L \geq \ell \geq 2$, a q-ary code C of blocklength n is said to be* (α, ℓ, L)-list recoverable *if given arbitrary "lists" $L_i \subseteq \mathbb{F}_q$ of size at most ℓ for each i, $1 \leq i \leq n$, the number of codewords $\mathbf{c} = \langle c_1, \ldots, c_n \rangle \in C$ such that $c_i \in L_i$ for at least αn values of i, is at most L.*

We will loosely refer to the task of decoding a code under the above model as "list recovering" the code.

Remark: A code of blocklength n is $(\alpha, 1, L)$-list recoverable if and only if it is $((1-\alpha)n, L)$-list decodable.

9.3 Pseudolinear Codes: Existence Results and Properties

In this section, we prove existence results using the probabilistic method for codes which serve as inner codes in our concatenated code constructions. The inner codes will be "pseudolinear codes" with appropriate parameters. We now formally define the notion of "pseudolinear" code families and prove some of the basic list decodability properties offered by random pseudolinear codes. An informal description of pseudolinear codes was given in Chapter 2, where we had put off a more detailed treatment to later when the machinery is really used (which is in this chapter).

The notion of pseudolinear codes plays a crucial role in translating list decodability results for general, non-linear codes into similar results for codes, which albeit not linear, still have a succinct description, and allow for efficient encoding. In our applications, these pseudolinear codes, which are typically used as inner codes in suitable concatenated schemes, are critical in getting efficient constructions for our codes.

9.3.1 Pseudolinear (Code) Families

Informally, an L-wise independent code family is a sample space of codes such that the encodings of any L non-zero messages are completely independent for a random code drawn from the family. The formal definition follows.

Definition 9.2. *An L-wise independent $(n,k)_q$-code family $\mathcal{F}$ is a sample space of codes that map k symbols over $\mathbb{F}_q$ to n symbols over $\mathbb{F}_q$ such that for every set of L non-zero messages $\mathbf{x_1}, \mathbf{x_2}, \ldots, \mathbf{x_L} \in \mathbb{F}_q^k$, the random variables $C(\mathbf{x_1}), C(\mathbf{x_2}), \ldots, C(\mathbf{x_L})$ are completely independent, for a code C picked uniformly at random from the family $\mathcal{F}$.*

A random code picked from an L-wise independent family often tends to have very good list decoding properties for decoding with list size L, owing to the mutual independence of any set of L (non-zero) codewords. An example of an L-wise independent code family is the space of all general, non-linear q-ary codes of blocklength n and dimension k, which is clearly L-wise independent, for all L, $1 \leq L < q^k$. While a random, non-linear code has excellent randomness properties, it comes from a very large sample space and there is no succinct representation of a general code from the family.[4] We now define a family of codes which we call *pseudolinear* that has the desired L-wise independence property and in addition is succinct. Thus a random code drawn this family has the desired randomness properties, can be succinctly represented, and has an efficient encoding procedure.

Definition 9.3 (Pseudolinear Codes). *For a prime power q, integer $L \geq 1$, and positive integers k, n with $k \leq n$, an (n,k,L,q)-pseudolinear family $\mathcal{F}(n,k,L,q)$ of codes is defined as follows. Let H be the parity check matrix of any q-ary linear code of blocklength $(q^k - 1)$, minimum distance at least $(L+1)$ and dimension $q^k - 1 - O(kL)$ (for example, one can use parity check matrices of q-ary BCH codes of designed distance $(L+1)$, cf. [10, Chap. 15]). A random code C_A in the pseudolinear family $\mathcal{F}(n,k,L,q)$ is specified by a random $n \times O(kL)$ matrix A over $\mathbb{F}_q$. Under the code C_A, a message $x \in \mathbb{F}_q^k \setminus \{0\}$ is mapped to $A \cdot H_x \in \mathbb{F}_q^n$ where $H_x \in \mathbb{F}_q^{O(kL)}$ is the column of*

[4] The space of random $[n,k]_q$ linear codes has the desired succinctness properties, but however is in general not even 3-wise independent (it is 2-wise (or pairwise) independent, though). This is because for any linear map $E : [q]^k \to [q]^n$, we have $E(x+y) = E(x) + E(y)$ for every $x, y \in [q]^k$.

H indexed by x (viewed as an integer in the range $[1, q^k)$). (We also define $H_0 = \mathbf{0}$ to be the all-zeroes vector.)

Given $1 \leq x < q^k$, a description of the column H_x can be obtained in time polynomial in k and $\log q$, since there are explicit descriptions of the parity check matrices of BCH codes of distance at least $(L+1)$ and blocklength $(q^k - 1)$, in terms of the powers of the generating element of $\mathrm{GF}(q^k)$ over $\mathrm{GF}(q)$ (see, for example, [132, Chap. 9]). Hence encoding as per these codes is an efficient operation. In addition to these complexity issues, the crucial combinatorial property about these pseudolinear codes that we exploit is that every set of L fixed non-zero codewords of the code C_A, for a random A, are completely independent. This is formalized in Lemma 9.4 below. Note also that, unlike general non-linear codes, codes from a pseudolinear family have a succinct representation, since they can be specified using the $n \times O(kL)$ "generator" matrix A and $\mathrm{poly}(k, \log q)$ sized information about the generating element of $\mathrm{GF}(q^k)$ over $\mathrm{GF}(q)$.

Lemma 9.4. *For every n, k, L, q, an (n, k, L, q)-pseudolinear family is an L-wise independent $(n, k)_q$ family of codes.*

Proof: Since H defines the parity check matrix of a code, say C, that has distance at least $(L+1)$, every set of L columns of H are linearly independent. Indeed, suppose this were not the case. Then there must exist a linear dependence $\alpha_1 H_{a_1} + \ldots + \alpha_L H_{a_L} = \mathbf{0}$ for integers $1 \leq a_1 < a_2 < \cdots < a_L < q^k$ and $\alpha_i \in \mathbb{F}_q$ with not all $\alpha_i = 0$. This implies that the non-zero vector $\mathbf{y}$ which has symbol α_i at location a_i for $i = 1, 2, \ldots, L$ and zeroes at all other locations satisfies $H \cdot \mathbf{y} = \mathbf{0}$ and hence belongs to the code C. But the Hamming weight of $\mathbf{y}$ is at most L, a contradiction to the fact that the distance of C is at least $(L+1)$.

Now consider any L non-zero codewords corresponding to messages $a_1, a_2, \ldots, a_L$ where each $a_i \in [1, q^k)$. They are encoded into the codewords $\mathbf{c_i} = A \cdot H_{a_i}$. Since the various H_{a_i} are linearly independent, for a random matrix A, the various $\mathbf{c_i}$'s are completely independent. This follows from the general fact that the images of a set $S = \{\mathbf{v_1}, \ldots, \mathbf{v_L}\}$ of linearly independent vectors in $\mathbb{F}_q^m$, under a linear transformation defined by a random $n \times m$ matrix A, are completely independent. This fact is easy to prove since the $\mathbf{v_i}$'s, being linearly independent, can be mapped by an invertible linear map into the standard basis vectors $\mathbf{e_1}, \mathbf{e_2}, \ldots, \mathbf{e_L}$, and the mutual independence of $A \cdot \mathbf{e_1}, A \cdot \mathbf{e_2}, \ldots, A \cdot \mathbf{e_L}$ for a completely random A is obvious. □

Remark: We note here that one of the standard constructions of d-wise independent binary random variables (say, $X_1, X_2, \ldots, X_n$) uses arguments similar to the above (cf. [10, Chap. 15], [107]). It also proceeds by the construction of a set $S \subseteq \{0,1\}^a$, where $a = O(d \log n)$, consisting of n vectors with the property that any subset of d vectors in S are linearly independent. The set S is picked to be the columns of a parity check matrix of a binary

code of blocklength n, dimension $(n-a)$, and minimum distance at least $(d+1)$. The random variable X_i is defined by picking a random vector in $\{0,1\}^a$ and taking its dot product with the i'th vector in S. The fact that any d of the vectors in S are linearly independent translates into the d-wise independence of the X_i's. Using parity check matrices of appropriate BCH codes, gives d-wise independent sample spaces of $O(n^{\lfloor d/2 \rfloor})$ size. This size is in fact optimal, up to a constant factor, cf. [10, Chap. 15].

We next define the notion of an infinite family of (L,q)-pseudolinear codes of increasing blocklength. Since we are interested in the asymptotic performance of codes, we will be interested in such code families of a certain rate.

Definition 9.5. *An infinite family of* (L,q)-pseudolinear codes $\mathcal{C}_{L,q}$ *is obtained by picking codes* $\{C_{A_i}\}_{i\geq 1}$ *of blocklengths* n_i *(with* $n_i \to \infty$ *as* $i \to \infty$*) where* C_{A_i} *belongs to the* (n_i, k_i, L, q)*-pseudolinear family.*

9.3.2 Probabilistic Constructions of Good, List Decodable Pseudolinear Codes

We now analyze the list decodability properties of random pseudolinear codes and use it to prove the existence of pseudolinear codes with a certain tradeoff between rate and list decodability. We stress that all existential results of this section are in fact "high probability results"; in other words, a random pseudolinear code with appropriate parameters achieves the claimed rate and list decodability properties with $(1-o(1))$ probability. We will use this fact implicitly when we use the codes guaranteed by this section in later (probabilistic) code constructions.

Lemma 9.6. *For every prime power* $q \geq 2$*, every integer* ℓ*,* $1 \leq \ell \leq q$ *and* $L \geq \ell$*, and every* α*,* $0 < \alpha \leq 1$*, there exists an infinite family of* $(L+1,q)$*-pseudolinear codes of rate* r *given by*

$$r \geq \frac{1}{\lg q} \min \left\{ \alpha \lg(q/\ell) - H(\alpha) - H(\ell/q) \cdot \frac{q}{L+1} \ , \right. \tag{9.1}$$
$$\left. \big(\alpha \lg(q/\ell) - H(\alpha)\big) \cdot \frac{L+1}{L} - H(\ell/q) \cdot \frac{q}{L} \right\} - o(1) \ ,$$

such that every code in the family is (α, ℓ, L)*-list recoverable. (Recall that for* $0 \leq x \leq 1$*,* $H(x) = -x \lg x - (1-x)\lg(1-x)$ *denotes the binary entropy function of* x*).*

Proof: The proof follows by employing the probabilistic method. Let n be large enough and r be as in the statement of the lemma. We will show that a code C picked at random from an $(n, rn, L+1, q)$-pseudolinear family is (α, ℓ, L)-list recoverable with high probability.

Let us estimate the probability that the code C is not (α, ℓ, L)-list recoverable. Fix a choice of $\mathcal{L}_i$, $1 \leq i \leq n$, where each $\mathcal{L}_i$ is a subset of $[q]$ of size

ℓ. We wish to bound from above the probability that for some set $S \subseteq C$ of size $L+1$, the event E_S that each codeword in S has some element from $\mathcal{L}_i$ in its i'th coordinate for at least αn values of i, occurs. We divide this event into two cases: (i) when the set S does not contain the zero codeword, and (ii) when the zero codeword belongs to S. In case (i), the probability of E_S is clearly at most

$$\left(\binom{n}{\alpha n}\left(\frac{\ell}{q}\right)^{\alpha n}\right)^{L+1} . \tag{9.2}$$

For case (ii), the probability is 0 unless at least αn of the $\mathcal{L}_i$'s include 0, in which case the probability is at most

$$\left(\binom{n}{\alpha n}\left(\frac{\ell}{q}\right)^{\alpha n}\right)^{L} . \tag{9.3}$$

By a union bound, the probability that for some choice of $\mathcal{L}_i$'s, some bad event E_S happens is at most

$$\binom{q}{\ell}^n q^{rn(L+1)}\left(\binom{n}{\alpha n}\left(\frac{\ell}{q}\right)^{\alpha n}\right)^{L+1} + \binom{n}{\alpha n}\binom{q-1}{\ell-1}^{\alpha n}\binom{q}{\ell}^{n-\alpha n} q^{rnL}\left(\binom{n}{\alpha n}\left(\frac{\ell}{q}\right)^{\alpha n}\right)^{L}$$

which is at most

$$\binom{q}{\ell}^n\left(q^{rn}\cdot 2^{H(\alpha)n}2^{-\lg(q/\ell)\alpha n}\right)^{L+1} + \binom{q}{\ell}^n \cdot q^{rnL}\cdot 2^{H(\alpha)n(L+1)}\cdot 2^{-\lg(q/\ell)\alpha n(L+1)} .$$

The above quantity is easily seen to be $o(1)$ for r as in Equation (9.1). Hence there is a $(1-o(1))$ probability that the code C has the claimed list recoverability properties. □

Corollary 9.7. *Let $\alpha > 1$ be an arbitrary constant. Then there exist positive constants a_α, b_α such that for every $\varepsilon > 0$, there exist $q = O(1/\varepsilon^2)$, $L = a_\alpha/\varepsilon$ and a family of (L,q)-pseudolinear codes of rate b_α which is $(\alpha, 1/\varepsilon, L)$-list recoverable.*

Proof: Follows by a straightforward substitution of $\ell = 1/\varepsilon$ and $q = O(1/\varepsilon^2)$ in the bound of Equation (9.1). □

We now obtain the following results for the "usual" notion of list decodability.

Lemma 9.8. *For every prime power $q \geq 2$, every p, $0 < p < 1$, and every integer $L \geq 2$, there exists an infinite family $\mathcal{C}_{L,q}$ of (L,q)-pseudolinear codes of rate r given by*

$$r \geq 1 - H_q(p) - \frac{1}{L} - o(1) ,$$

such that $\mathsf{LDR}_L(\mathcal{C}_{L,q}) \geq p$.

Proof: The proof follows by an application of the probabilistic method similar to that of Lemma 9.6. Let us pick a code C at random from an (n, rn, L, q)-pseudolinear family where n is large enough and r is as in the statement of the lemma. Let us estimate the probability that C is *not* (pn, L)-list decodable. In this case there must be some L non-zero codewords of C all of which lie within a Hamming ball of radius pn. Since any L non-zero codewords of C are mutually independent, the probability of this happening for a fixed Hamming ball $B_q(\mathbf{x}, pn)$ is at most

$$\binom{q^{rn}-1}{L} \cdot \Big(\frac{q^{H_q(p)n}}{q^n}\Big)^L$$

since $|B_q(\mathbf{x}, pn)| \leq q^{H_q(p)n}$. The probability that this happens for *some* $\mathbf{x} \in [q]^n$ is thus at most

$$q^n q^{rnL} q^{(H_q(p)-1)Ln}$$

which is $o(1)$ for r as in the statement of the lemma. Hence a random pseudolinear code of rate r is (pn, L)-list decodable with high probability. □

Remark: It is possible to state a more complicated bound, similar to that of Lemma 9.6, by analyzing separately the cases when the "bad" set of $(L+1)$ codewords contains the zero codeword and when it doesn't. For simplicity, and since the above bound is all that we will need, we just stated and proved that above. For Lemma 9.6, the more careful argument has the advantage of showing that positive rate is possible even when $L = \ell$ for large enough α, which is why we stated the more complicated bound.

Corollary 9.9. *Let $a > 1$ be an arbitrary constant. Then there exist constants $b_a, c_a > 1$ such that for every $\varepsilon > 0$ the following holds: let $q = O(1/\varepsilon^a)$ and $L = b_a/\varepsilon$. Then there exists a rate ε/c_a family of (L, q)-pseudolinear codes PL_ε which satisfies $\mathsf{LDR}_L(\mathrm{PL}_\varepsilon) \geq 1 - \varepsilon$.*

List Recoverability of Random Linear Codes We now state the version of Lemma 9.6 that applies to random *linear* codes. This can be viewed as the generalization of Theorem 5.6 (from Chapter 5), which analyzed the list decodability of random linear codes, to the list recoverability situation. The result for linear codes will be used in the multi-concatenated code construction in Section 9.5 (specifically in the proof of Lemma 9.21).

Lemma 9.10. *For every prime power $q \geq 2$, every integer ℓ, $1 \leq \ell \leq q$ and $L > \ell$, and every α, $0 < \alpha \leq 1$, there exists an infinite family of linear codes of rate r given by*

$$r \geq \frac{1}{\lg q}\Big(\alpha \lg(q/\ell) - H(\alpha) - H(\ell/q) \cdot \frac{q}{\log_q(L+1)}\Big) - o(1)\ , \tag{9.4}$$

such that every code in the family is (α, ℓ, L)-list recoverable.

Proof: The proof follows along the lines of Lemma 9.6 by analyzing the performance of a linear code defined by a random $(n \times rn)$ generator matrix over $\mathbb{F}_q$. If some set of $(L+1)$ codewords violate the (α, ℓ, L)-list recoverability property, then there must be at least $\log_q(L+1)$ non-zero codewords among them that correspond to encodings of linearly independent messages in $\mathbb{F}_q^{rn}$. It therefore suffices to prove an upper bound on the probability that some set of $\log_q(L+1)$ *linearly independent* messages are mapped into codewords that violate the $(\alpha, \ell, \log_q(L+1))$-list recoverability property. Since, for a random linear code the codewords associated with a set of linearly independent messages are all *mutually independent* (cf. Lemma 9.4), the analysis of Lemma 9.6 goes through with $\log_q(L+1)$ taking the place of $L+1$ in Equation (9.2). The claimed bound then follows. □

Corollary 9.11. *Let $\alpha \leq 1$ be an arbitrary constant. Then there exist positive constants a_α, c_α such that for every $\varepsilon > 0$, there exist $q = O(1/\varepsilon^2)$, $L = q^{c_\alpha/\varepsilon}$ and a family of q-ary* linear *codes of rate a_α which is $(\alpha, 1/\varepsilon, L)$-list recoverable.*

9.3.3 Derandomizing Constructions of Pseudolinear Codes

One straightforward way to "constructivize" or "derandomize" the probabilistic result of Lemmas 9.6 and 9.8 is by a brute-force search over all codes in an (n, k, L, q)-pseudolinear family. Note that checking whether a fixed $(n, k)_q$ pseudolinear code has the necessary (α, ℓ, L)-list recoverability or (pn, L)-list decodability properties can be done by a search over all "received words" and over all codewords in $q^{O(\ell n+k)}$ and $q^{O(n)}$ time, respectively. However, going over all possible $(n, k)_q$ pseudolinear codes involves going over all $n \times O(kL)$ "generator" matrices and this requires $q^{O(knL)}$ time. Hence a naive derandomization of the probabilistic constructions of $(n, k)_q$ pseudolinear codes from the previous section will take $q^{O(knL)}$ time. This is prohibitive even for the blocklengths for inner codes. For example, if we wish to use a pseudolinear code as an inner code in a concatenation scheme with outer code being a Reed-Solomon code over a polynomially large field, then the dimension of the pseudolinear code will be logarithmic in the overall blocklength. The naive derandomization will take quasi-polynomial time in such a case, while we would clearly prefer a polynomial time deterministic construction. We next demonstrate how one can find codes in a (n, k, L, q)-pseudolinear family with the properties claimed in Lemmas 9.6 and 9.8 in $q^{O(kL+n\ell)}$ time. Applied to the above-mentioned concatenated code setting, this will enable polynomial time construction of the concatenated code, since ℓ, L, q will be constants and n, k will be logarithmic in the overall blocklength.

The basic idea is to derandomize the probabilistic constructions using the method of conditional expectations. Since the method is quite standard,

we only discuss informally how to apply it to our context.[5] We focus on Lemma 9.6, and the result for Lemma 9.8 is similar. To derandomize the result of Lemma 9.6, we will successively find the n rows of a "good" $n \times O(kL)$ matrix A such that the associated code C_A in the pseudolinear family $\mathcal{F}(n, k, L, q)$ is (α, ℓ, L)-list recoverable. (Here and in what follows $k = rn$ is the dimension of the code.)

Assume that for some $1 \leq s \leq n$, the first $(s-1)$ rows of A have been picked to be $\mathbf{a_1}, \ldots, \mathbf{a_{s-1}}$ where each $\mathbf{a_i} \in \mathbb{F}_q^{O(kL)}$. We pick $\mathbf{a_s}$ so that it minimizes a certain conditional expectation by searching among all the $q^{O(kL)}$ possible choices for $\mathbf{a_s}$.

The relevant expectation that we bound is the following. For each (ordered) collection $\mathcal{D}$ of n "lists" $\mathcal{L}_i \subseteq \mathbb{F}_q$ with $|\mathcal{L}_i| = \ell$ for each i, $1 \leq i \leq n$, each set of L (non-zero) codewords (given by a subset $T = \{x_1, \ldots, x_L\} \subseteq \mathbb{F}_q^k$ of size L) of the pseudolinear code, and each (ordered) collection $\mathcal{S}$ of L subsets $S_1, \ldots, S_L \subseteq [n]$ with each $|S_j| = \alpha n$, define an indicator random variable $I(\mathcal{S}, \mathcal{D}, T)$ as follows. $I(\mathcal{S}, \mathcal{D}, T)$ equals 1 if, for each j, $1 \leq j \leq L$, the codeword corresponding to $x_j \in T$ agrees with an element of $\mathcal{L}_i$ for each of the αn values of $i \in S_j$. Otherwise, $I(\mathcal{S}, \mathcal{D}, T) = 0$. In words, $I(\mathcal{S}, \mathcal{D}, T) = 1$ iff the setting $\mathcal{S}$, $\mathcal{D}$, T shows a "counterexample" to the code that we construct being (α, ℓ, L)-list recoverable (and is thus a "bad" event that we wish to avoid).

The random variable we consider in order to apply the method of conditional expectations is

$$X(\alpha, \ell, L) = \sum_{\mathcal{S}, \mathcal{D}, T} I(\mathcal{S}, \mathcal{D}, T) \ . \tag{9.5}$$

We will exploit linearity of expectation to compute the conditional expectations of $X(\alpha, \ell, L)$. The initial expectation of each $I(\mathcal{S}, \mathcal{D}, T)$ (taken over the random choice of all rows $\mathbf{r_i}$ of A, where $1 \leq i \leq n$) clearly equals $\left(\frac{\ell}{q}\right)^{\alpha n L}$ since the events for the various codewords in T are independent (by the L-wise independence property of the code). Multiplying this by the number of choices of $\mathcal{S}, \mathcal{D}, T$, we get (as in the proof of Lemma 9.6) that the initial expectation of $X(\alpha, \ell, L)$ is exponentially small (and in particular there exists a code with $X(\alpha, \ell, L) = 0$, or in other words which is (α, ℓ, L)-list recoverable).

Once we condition on the first s rows of A being fixed to, say, $\mathbf{a_1}, \ldots, \mathbf{a_s}$, the expected value of $I(\mathcal{S}, \mathcal{D}, T)$ taken over the random choices of the remaining $(n-s)$ rows $\mathbf{r_1}, \ldots, \mathbf{r_{n-s}}$ can still be exactly computed. Indeed, the first s coordinates of each of the codewords corresponding to each $x_j \in T$, for $1 \leq j \leq L$, are now fixed, and one can compute for each of them the number of coordinates in $S_j \cap \{1, 2, \ldots, s\}$ for which the codeword agrees with an element from the associate list $\mathcal{L}_i$. Thus, for each x_j, we can exactly compute

[5] The reader can find a discussion of the method of conditional expectations, for example, in [10, Chap. 15].

the probability that the associated codeword will agree with an element from $\mathcal{L}_i$ for each $i \in S_j$ when the remaining rows are picked at random. By the L-wise independence property, we can then simply multiply the probabilities for the various x_j's to estimate the conditional expectation of $I(\mathcal{S}, \mathcal{D}, T)$. We can do so for each of $\binom{n}{\alpha n}^L \cdot \binom{q}{\ell}^n \cdot \binom{q^k-1}{L}$ choices of $(\mathcal{S}, \mathcal{D}, T)$, and then add up all these expectations to exactly compute the conditional expectation of X. This is of course sufficient to make the best choice for $\mathbf{a_s}$, given that $\mathbf{a_1}, \ldots, \mathbf{a_{s-1}}$ have already been picked. Once we pick $\mathbf{a_i}$, for $1 \leq i \leq n$, we have the required pseudolinear code that satisfies the property of Lemma 9.6.

Applying the above to the case $q = O(1/\varepsilon^2)$, $\ell = 1/\varepsilon$ and $L = O(1/\varepsilon)$, one can thus prove the following lemma, which is one of the main results we need for our later constructions. This is the "constructive" version of Corollary 9.7. The alphabet size q can actually be made $O(1/\varepsilon^c)$ for any $c > 1$, but since this will not be important to us, we state the result for an alphabet size which is at least $\Omega(1/\varepsilon^2)$. The claims about the representation size and encoding follow since any member of an (n, k, L, q)-pseudolinear family can be represented by an $n \times O(kL)$ matrix over $\mathbb{F}_q$ and encoding involves multiplying a vector in $\mathbb{F}_q^{O(kL)}$ with this matrix. The lower bound claimed on the rate follows from Equation (9.1) after a simple calculation.

Lemma 9.12. *For every α, $0 < \alpha < 1$, and for all large enough constants $c > 1$, there exists a positive constant $a_\alpha \geq \frac{1}{3}(\alpha - 1/c)$ such that for all small enough $\varepsilon > 0$ the following holds. For all prime powers $q = \Omega(1/\varepsilon^2)$, there exist $L = c/\varepsilon$ and a family $\mathrm{PL}_\varepsilon^{(1)}$ of (L, q)-pseudolinear codes of rate a_α, such that a code of blocklength n in the family has the following properties:*

- *(a) it is $(\alpha, 1/\varepsilon, L)$-list recoverable,*
- *(b) it is constructible in deterministic time $q^{O(n\varepsilon^{-1})} = 2^{O(n\varepsilon^{-1} \log q)}$ or with high probability in randomized $O(n^2\varepsilon^{-1} \log q)$ time (i.e., the constructed code will have the list recoverability property claimed in (a) with high probability), and*
- *(c) it can can be represented in $O(n^2\varepsilon^{-1} \log q)$ space, and encoded using $O(n^2\varepsilon^{-2})$ operations over $\mathbb{F}_q$.*

We also get the following constructive version of Corollary 9.9 by applying the same derandomization procedure.

Lemma 9.13. *Let $a > 1$ be an arbitrary constant. Then there exist constants $b_a, c_a > 1$ such that for every $\varepsilon > 0$ the following holds. For all prime powers $q = \Omega(1/\varepsilon^a)$, there exist $L = b_a/\varepsilon$ and a family $\mathrm{PL}_\varepsilon^{(2)}$ of (L, q)-pseudolinear codes of rate at least ε/c_a, such that a code of blocklength n in the family has the following properties:*

(a) it is $((1-\varepsilon)n, L)$*-list decodable.*
(b) it is constructible in deterministic time $q^{O(n)} = 2^{O(n \log q)}$*, or with high probability in randomized* $O(n^2 \log q)$ *time, and*
(c) it can can be represented in $O(n^2 \varepsilon^{-1} \log q)$ *space, and encoded using* $O(n^2 \varepsilon^{-2})$ *field operations over* $\mathbb{F}_q$*.*

A similar result for linear codes. We now state a result analogous to Lemma 9.12 for the case of linear codes.

Lemma 9.14. *For every constant* α*,* $0 < \alpha < 1$*, and for all large enough constants* $c > 1$*, there exists a positive constant* $a_\alpha \geq \frac{1}{3}(\alpha - 1/c)$ *such that for all small enough* $\varepsilon > 0$ *the following holds. For all prime powers* $q = \Omega(1/\varepsilon^2)$*, there exists a family of* q*-ary linear codes of rate* a_α*, such that a code of blocklength* n *in the family has the following properties:*

(a) it is $(\alpha, 1/\varepsilon, q^{c/\varepsilon})$*-list recoverable,*
(b) it is constructible in deterministic time $q^{O(n\varepsilon^{-1})}$*, or with high probability in randomized* $O(n^2 \log q)$ *time, and*
(c) it can can be represented in $O(n^2 \log q)$ *space, and encoded using in* $O(n^2)$ *operations over* $\mathbb{F}_q$*.*

Proof: The claimed parameters follow by substituting $\ell = 1/\varepsilon$, $q = \Omega(1/\varepsilon^2)$, and $L = q^{c/\varepsilon}$ in Lemma 9.10. Note that since the code is linear, it can be represented using its generator matrix, which takes $O(n^2)$ entries in $\mathbb{F}_q$. The only non-trivial thing to check is the claimed deterministic construction time. A naive derandomization will involve trying out all $(n \times a_\alpha n)$ generator matrices, and this will take $q^{O(n^2)}$ time (the verification of the list recoverability property can be done in $q^{O(n\varepsilon^{-1})}$ time). However, as in the case of pseudolinear codes, one can use the method of conditional expectations to get a faster derandomization of the probabilistic construction of Lemma 9.10. This will involve picking the n rows of the generator matrix in sequence, each time searching for the best row from $\mathbb{F}_q^{a_\alpha n}$ that minimizes a certain conditional expectation. The relevant conditional expectations can be computed in $q^{O(n\varepsilon^{-1})}$ time. Hence, the total time required to find a generator matrix that defines a code with the required properties is $q^{O(n\varepsilon^{-1})}$. We omit the details which are very similar to the derandomization of the pseudolinear case. □

Remark concerning alphabet size. Even though the above results are stated for code families over a fixed constant-sized alphabet, a variant of it holds equally well also for alphabet size that grows with the length of the code (in some sense the large alphabet only "helps" these results; note also that the statements of Lemmas 9.12, 9.13, and 9.14 only pose lower bounds on q). This fact is later exploited in our multi-concatenated code constructions from Section 9.5, where we shall make use of such codes for q which is of the form 2^{n^p} for some integer p (n being the blocklength of the code). It

is also used in the next section where we show how pseudolinear codes over such large alphabets can be decoded in time significantly better than a brute-force search over all codewords. It is in fact this construction that is used in Section 9.5.

Remark concerning "density" of the codes in the families. Since the existence results claimed in the previous several lemmas are proved by a straightforward application of the probabilistic method, it follows that there exist such codes with *any* (large enough) dimension one seeks (and the properties such as rate and list decodability stay as claimed in the lemmas). We do not explicitly state this fact in the results, but the result of the next section is conveniently stated by fixing the dimension, and hence we explicitly state that it achieves any desired dimension for the codes it constructs. In its proof, as well as in other proofs, we will implicitly use that this fact also holds for the codes from the several previous lemmas.

9.3.4 Faster Decoding of Pseudolinear Codes over Large Alphabets

The naive algorithm to $(\alpha, 1/\varepsilon, O(1/\varepsilon))$-list recover the pseudolinear codes from Lemma 9.12 is to simply run over all the $q^{O(n)}$ codewords and output only those which satisfy the list recoverability requirement. This takes $q^{\Omega(n)}$ time. In Section 9.5, we will use pseudolinear codes over large alphabets (exponential in the blocklength) in a multi-concatenated scheme, in a hope of getting sub-exponential decoding algorithms for the final code that we construct. But the $q^{\Omega(n)}$ runtime for the decoding is prohibitive for such an application due to the huge value of q.

We now present a code construction that combines pseudolinear codes along with a "parallel" encoding by a linear code to improve the decoding time for codes over very large alphabets. For want of a better term, we refer to these codes as "large alphabet pseudolinear codes". Each symbol of the final encoding will be the "juxtaposition" of the symbols corresponding to the linear and pseudolinear encodings. The linear component of the encoding will be list recoverable in much faster time than the pseudolinear code. The exact details appear in Lemma 9.15 below. The codes constructed below will be the ones that are used in Section 9.5. The technique of symbol juxtaposition used here will be again used in Section 9.6 of this chapter, and in the next chapter on list decodable erasure codes. We believe that just like pseudolinear codes, it is also an important code design tool to take home from this chapter.

Lemma 9.15. *For every constant α, $0 < \alpha < 1$, and all sufficiently large constants $c > 1$, there exists a constant $b_\alpha \leq 6\big(\alpha - 1/c\big)^{-1}$ such that $\forall\ \varepsilon > 0$ there exists $q = O(1/\varepsilon^2)$ for which the following holds. For all integers m, s, there exists a code of dimension m and blocklength at most $b_\alpha m$ over $\mathrm{GF}(q^{2s})$ with the following properties:*

(i) It is $(\alpha, 1/\varepsilon, c/\varepsilon)$-list recoverable in $O(s^3(1/\varepsilon)^{O(m)})$ time.

(ii) It is constructible deterministically in $q^{O(sm\varepsilon^{-1})} = 2^{O(sm\varepsilon^{-1}\log q)}$ time. A probabilistic construction that has the claimed list recoverability property with high probability can be found in $O(m^2(s\varepsilon^{-1} + s^2)\log q)$ time. The code can be encoded in $O(m^2 s^2 \log^2 q)$ time.

Proof: Let $\varepsilon > 0$ be given, and let $q = O(1/\varepsilon^2)$ be a power of two. By Lemma 9.12, we know that for every α and all large enough c, there exists a pseudolinear code over $\mathrm{GF}(q^s)$, say C_1, of dimension $2m$, blocklength $b_\alpha m$ such that C_1 is $(\alpha, 1/\varepsilon, c/\varepsilon)$-list recoverable and is constructible in $2^{O(ms\varepsilon^{-1}\log q)}$ deterministic time or in $O(m^2 s\varepsilon^{-1}\log q)$ probabilistic time. Note that we may assume that $b_\alpha \leq 6(\alpha - 1/c)^{-1}$ since the rate of the codes guaranteed by Lemma 9.12 is at least $\frac{1}{3}(\alpha - 1/c)$.

The only known list recovering algorithm for such a pseudolinear code is to perform a brute-force search over all $(q^s)^{2m}$ possible codewords, which takes $(1/\varepsilon)^{O(ms)}$ time. In order to speed up the algorithm, we perform an encoding with a suitable random linear code in *parallel* — each symbol of the final encoding will be the "juxtaposition" of the symbols corresponding to the linear and pseudolinear encodings. The linear code, say C_{lin}, will be a q-ary code of dimension $2ms$ and blocklength $b_\alpha ms$ which is $(\alpha, 1/\varepsilon, q^{c/\varepsilon})$-list recoverable. By the result of Lemma 9.14, such a linear code exists and can be constructed deterministically in $q^{O(ms\varepsilon^{-1})}$ time, or probabilistically in $O(m^2 s^2 \log q)$ time.

By "aggregating" each set of successive s symbols in both the message and its encoding by C_{lin}, we can view C_{lin} as a code over $\mathrm{GF}(q^s)$. Viewed this way, C_{lin} will map $2m$ symbols over $\mathrm{GF}(q^s)$ into $b_\alpha m$ symbols over $\mathrm{GF}(q^s)$. To avoid confusion, let us denote the code C_{lin} viewed as a code over $\mathrm{GF}(q^s)$ by $\tilde{C}_{\mathrm{lin}}$.[6]

We now claim that the resulting code $\tilde{C}_{\mathrm{lin}}$ is $(\alpha, 1/\varepsilon, q^{c/\varepsilon})$-list recoverable in

$$O((1/\varepsilon)^{O(m)} m^3 s^3 \log^{O(1)} q) = O(s^3(1/\varepsilon)^{O(m)})$$

time. The combinatorial list recoverability property itself follows since C_{lin} is $(\alpha, 1/\varepsilon, q^{c/\varepsilon})$-list recoverable as a code over $\mathrm{GF}(q)$, and the property therefore definitely holds for the code $\tilde{C}_{\mathrm{lin}}$ obtained by viewing C_{lin} as a code over $\mathrm{GF}(q^s)$. To prove the claim about the time complexity for list recovering $\tilde{C}_{\mathrm{lin}}$, we present the following algorithm. The algorithm is simply to try out all possible subsets S of $\alpha b_\alpha m$ positions and for each such choice go over all possible sets T of $(1/\varepsilon)^{O(m)}$ symbols from the input lists. For each such choice of S and T, we find if any codeword of C_{lin} is consistent with these symbols (this is simply an erasure decoding of the linear code). This can be done by solving a linear system over $\mathrm{GF}(q)$ and takes at most $O((2ms)^3 \log^{O(1)} q)$

[6] The code $\tilde{C}_{\mathrm{lin}}$ will not in general be linear over $\mathrm{GF}(q^s)$, but we will only use linearity of C_{lin} over $\mathrm{GF}(q)$.

time since the blocklength of C_{lin} equals $2ms$. Since C_{lin} is $(\alpha, 1/\varepsilon, q^{c/\varepsilon})$-list recoverable, it is definitely true that the number of codewords of C_{lin} consistent with a certain choice of symbols in a fraction α of the positions is at most $q^{c/\varepsilon}$. Finally, we will have to check which of the codewords of C_{lin} actually yield codewords of $\tilde{C}_{\text{lin}}$ that meet the required list recoverability condition. Since each erasure decoding yields at most $q^{c/\varepsilon}$ solutions to check, the total runtime will be the number of choices of S, T multiplied by the time for each erasure decoding of C_{lin}, plus an additional time of roughly $O(q^{c/\varepsilon})$ to prune the list returned by the erasure decoding of C_{lin} . This gives the claimed $O((1/\varepsilon)^{O(m)} s^3)$ runtime.

We now define our final code C^* to be the *juxtaposition* of C_1 and $\tilde{C}_{\text{lin}}$; i.e. to encode a message according to C^*, we encode it using C_1 and $\tilde{C}_{\text{lin}}$ independently to get two strings, say, $\langle a_1, a_2, \ldots, a_t \rangle$ and $\langle b_1, b_2, \ldots, b_t \rangle$, where $t = b_\alpha m$ and each $a_i, b_i \in \text{GF}(q^s)$. The encoding of that message as per C^* will then be $\langle c_1, \ldots, c_t \rangle$, where each $c_i = (a_i, b_i)$ is viewed as an element of $\text{GF}(q^{2s})$. Note that C^* defined this way encodes $2m$ symbols over $\text{GF}(q^s)$ into $b_\alpha m$ symbols over $\text{GF}(q^{2s})$. We may equivalently view C^* as mapping m symbols over $\text{GF}(q^{2s})$ into $b_\alpha m$ symbols over $\text{GF}(q^{2s})$. In other words C^* has dimension m and blocklength $b_\alpha m$ as a q^{2s}-ary code.

Since C_1 is $(\alpha, 1/\varepsilon, c/\varepsilon)$-list recoverable, so is C^* (as would any juxtaposed code that involves C_1). This gives the combinatorial list recoverability property of C^*. To obtain the claim about the algorithmic list recoverability, we will use the "linear" component $\tilde{C}_{\text{lin}}$ of C^*. By the above argument, $\tilde{C}_{\text{lin}}$ can be $(\alpha, 1/\varepsilon, q^{c/\varepsilon})$-list recovered within the claimed runtime. One can then run through the at most $q^{O(1/\varepsilon)}$ messages output by this algorithm and "cross-check" if its encoding by the pseudolinear code C_1 agrees with the respective component of the symbols in the input lists on a fraction α of the positions. By the combinatorial list recoverability property of C_1, at most c/ε of the messages will pass this check. These will be the messages output by the algorithm. The running time of this procedure is dominated by that of the list recovering algorithm for $\tilde{C}_{\text{lin}}$, and is thus $O((1/\varepsilon)^{O(m)} s^3)$.

The encoding time for C^* is dominated by the time to encode the "linear" component. Since the code C_{lin} has both dimension and blocklength at most $O(ms)$, the encoding of C_{lin} takes at most $O((ms)^2 \log^2 q)$ time. This completes the proof of the lemma. □

9.4 The Basic Expander-Based Construction of List Decodable Codes

For the construction in this section, we will use families of graphs with a small bounded degree which nevertheless have strong connectivity properties. Specifically, they will have the "dispersing" property that the neighborhood of any large enough subset of vertices misses a very small fraction of vertices. We mention here that code constructions in Chapter 11 are obtained using

similar techniques and also use expander graphs, but there we will make use of stronger properties than the above dispersion property — namely we will use certain isoperimetric properties offered by expander graphs (which will be discussed in 11.2). We next define the specific expansion property we need for the results of this chapter.

9.4.1 Definition of Required "Expanders"

Definition 9.16. *For integers $N, d \geq 1$ and $0 < \varepsilon, \alpha < 1$, an $(N, d, \alpha, \varepsilon)$-*disperser *is a d-regular $N \times N$ bipartite graph $H = (A, B, E)$ (where A, B with $|A| = |B| = N$ are the two sets in the bipartition and E is the edge set), with the property that given any subset $X \subseteq B$ with $|X| \geq \varepsilon|B|$, the number of vertices in A with some neighbor in X is at least $\alpha|A|$.*

Disperser graphs were first defined by Sipser [170] and since then there has been a wide body of work on the properties and explicit constructions of dispersers (cf. the survey by Nisan [148]). The following result on the existence of disperser graphs is well known, see for instance [10, Chap. 9] (and also Section 11.2 of this book) where an explicit construction using the Ramanujan graphs of [131] is discussed.

Fact 9.17 *There is a constant c such that for every $\varepsilon > 0$ and for infinitely many n, there exists an explicitly constructible $(n, c/\varepsilon, 1/2, \varepsilon)$-disperser.*

Of course the $1/2$ in the above claim can be changed to any fixed constant $\alpha < 1$. In such a case, the constant c in the degree will depend on α.

9.4.2 Reduction of List Decoding to List Recoverability Using Dispersers

We now present an elegant and simple reduction of the problem of constructing codes which are efficiently $((1-\varepsilon)n, L)$-list decodable to the problem of constructing codes with efficient $(\alpha, O(1/\varepsilon), L)$-list recoverability, for some fixed constant α, say $\alpha = 1/2$. This idea is at the heart of all our expander-based code constructions that we present in this chapter. It is instructive to point out that the use of the expanders in our constructions is confined to this reduction, and the construction of good list recoverable codes itself is accomplished using other techniques.

The reduction is accomplished by redistributing the symbols of the codewords of a list recoverable code, say C_1, using a expander H, and thus define the codewords of a new code C_2 over a larger alphabet. The list recoverability property of C_1, together with the dispersion property of H, will imply the good list decodability of C_2. Given a corrupted received word $\mathbf{r}$ of C_2, one can push the symbols of $\mathbf{r}$ along the edges of the bipartite graph H to obtain a list of possible symbols for each position of C_1. The dispersion property

of H will imply that at least a fraction $1/2$ of these lists contain the correct symbol of the codeword of C_1. Now, the list recoverability property of C_1 can be used to complete the decoding. The formal statement of the reduction and the proof follow.

Proposition 9.18. *There exists an absolute constant c such that for every $\varepsilon > 0$ the following holds. Suppose there exists a q-ary code C_1 of blocklength n and rate r that is $(1/2, c/\varepsilon, L)$-list recoverable by an algorithm running in time $O(T(n))$. Further assume that n is such that there exists an $(n, c/\varepsilon, 1/2, \varepsilon)$-disperser. Then there exists a code C_2, which is explicitly specified given C_1, and which has the following properties:*

(i) It has blocklength n and rate $\varepsilon r/c$.
(ii) It is defined over an alphabet of size $q^{c/\varepsilon}$.
(iii) It is $((1-\varepsilon)n, L)$-list decodable, and moreover there is an algorithm to list decode C_2 up to a fraction $(1-\varepsilon)$ of errors in time $O(T(n) + n \log q/\varepsilon)$.

Proof: The code C_2 is obtained by distributing the symbols of codewords in C_1 using the edges of an $(n, \Delta, 1/2, \varepsilon)$-disperser where $\Delta = c/\varepsilon$. This is in a manner similar to Alon et al [6], who used such a symbol redistribution for the purpose of getting codes with a large (viz., $(1-\varepsilon)$) relative distance. Formally, let $H = ([n], [n], E)$ be an $(n, \Delta, 1/2, \varepsilon)$-disperser. For $1 \leq j \leq \Delta$ and $1 \leq i \leq n$, denote by $\Gamma_j(i)$ the j'th neighbor (on the left side) of the i'th vertex on the right side of H (we assume some fixed ordering of the neighbors of each node). A codeword $(c_1, c_2, \ldots, c_n)$ of C_1 is mapped into a codeword $(\tilde{c}_1, \ldots, \tilde{c}_n)$ of C_2, where each $\tilde{c}_i \in [q]^{\Delta}$ is given by $\tilde{c}_i = \langle c_{\Gamma_1(i)}, \ldots, c_{\Gamma_\Delta(i)} \rangle$. The claim about the blocklength, rate and alphabet size of C_2 follow immediately.

The algorithm for list decoding C_2 up to a radius of $(1-\varepsilon)n$ proceeds in two steps. Assume $\mathbf{r}$ is a received word and the goal is to find all codewords of C_2 that are within a Hamming distance of $(1-\varepsilon)n$ from $\mathbf{r}$. In other words, the goal is to find every message $\mathbf{x}$ that satisfies $\Delta(C_2(\mathbf{x}), \mathbf{r}) \leq (1-\varepsilon)n$. In the first step of the decoding, each position of the received word $\mathbf{r}$ "votes" on those positions of the corresponding codeword in C_1 which are adjacent to it in the disperser H. This gives for each i, $1 \leq i \leq n$, a list L_i of at most $\Delta = c/\varepsilon$ elements from $[q]$ for each position of the code C_1. See the illustration in Figure 9.3. In the second step, the $(1/2, c/\varepsilon, L)$-list recovering algorithm for C_1 is run with these lists L_i, $1 \leq i \leq n$, as input. Finally, for each message output by the list decoder for C_1, we check if its encoding under C_2 agrees with $\mathbf{r}$ in at least εn positions, and if so, we output it.

The time required for the above algorithm is the time for the first "voting" stage, which takes $O(n \log q/\varepsilon)$ time, followed by the time for list recovering C_1, which takes $O(T(n))$ time by hypothesis.

It remains to prove the correctness of the algorithm. Let $\mathbf{x}$ be any message such that $\Delta(C_2(\mathbf{x}), \mathbf{r}) \leq (1-\varepsilon)n$. Let $X \subseteq [n]$ be the set of positions where $C_2(\mathbf{x})$ and $\mathbf{r}$ agree. By hypothesis $|X| \geq \varepsilon n$. Define $Y \subseteq [n]$ to be the set of vertices on the left side of H which have a neighbor in X on the right. By

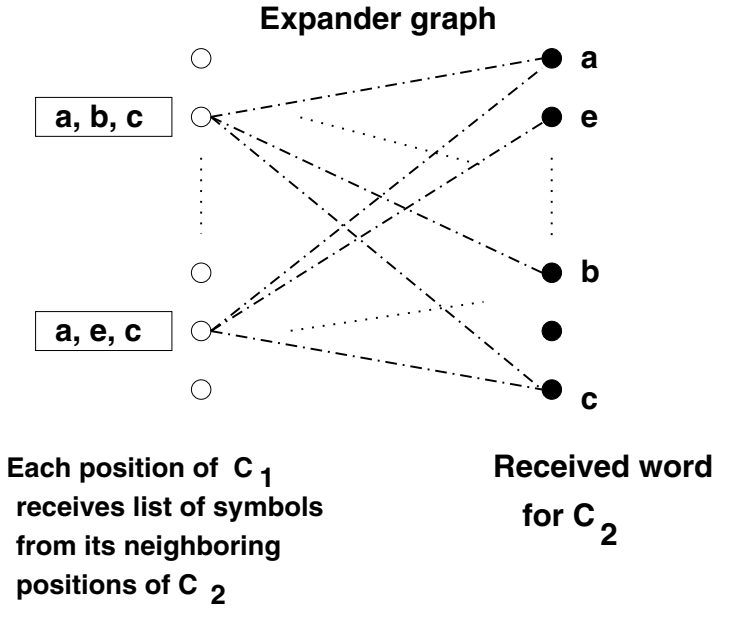

Fig. 9.3. Illustration of the decoding algorithm. Each position on the left collects a list of symbols from all its neighbors on the right. These lists are then used as input to the list recovering algorithm for the left code C_1. The dispersion property implies that even if the received word for C_2 had several errors, a good fraction of the lists obtained for C_1 contain the correct symbol.

the dispersion property of H, $|Y| \geq n/2$. Now, clearly for each $i \in Y$, the i'th symbol of $C_1(\mathbf{x})$ is included in the list L_i (since all votes coming from the positions in X are correct, and the symbols in Y are precisely those which receive at least one vote from the positions in X). Therefore, the message $\mathbf{x}$ will be included in the list output by the $(1/2, c/\varepsilon, L)$-list recovering algorithm for C_1, when it is run with the lists L_i as input. Hence, the above algorithm will successfully include $\mathbf{x}$ in the final list it outputs. □

The following states a more general form of the above proposition which states a stronger list recoverability property for C_2 using that of C_1. The proof is identical to the above — at the voting stage of decoding, instead of each position of C_2 passing one vote to each of its neighbors, it passes ℓ votes where ℓ is the number of possible symbols listed for that position. Proposition 9.18 follows with the setting $\ell = 1$.

Lemma 9.19. *There exists an absolute constant c such that for every $\varepsilon > 0$ the following holds. Suppose there exists a q-ary code C_1 of blocklength n and rate r that is $(1/2, c\ell/\varepsilon, L)$-list recoverable by an algorithm running in time*

$O(T(n))$. Further assume that n is such that there exists an $(n, c/\varepsilon, 1/2, \varepsilon)$-disperser. Then there is a code C_2, which is explicitly specified given C_1, with the following properties:

(i) It has blocklength n and rate $\varepsilon r/c$.
(ii) It is defined over an alphabet of size $q^{c/\varepsilon}$.
(iii) It is (ε, ℓ, L)-list recoverable, and moreover there is an algorithm to (ε, ℓ, L)-list recover C_2 in time $O(T(n) + n\ell \log q/\varepsilon)$.

9.4.3 Codes of Rate $\Omega(\varepsilon^2)$ List Decodable to a Fraction $(1 - \varepsilon)$ of Errors

We now present our code construction (number 1) which has rate ε^2 and is list decodable in near-quadratic time from up to a fraction $(1 - \varepsilon)$ of errors. The formal result is stated in Theorem 9.20.

Before we state and prove this result, we would like to point out one technical point concerning the constructions. Recall the overall structure of all our constructions (Figure 9.2): they use a certain "left code" C and then redistribute symbols of a codeword of C using an expander. There is an implicit assumption here that each side of the bipartite expander has the same number of vertices, say n, as the blocklength of C. The known constructions of Ramanujan graphs (eg. [131, 136]) work for infinitely many values of n, but not for *all* sufficiently large n (as would be ideal for our application). However, as discussed in [175, Section 2.4.1], these constructions give a *dense* sequence of graphs, i.e., the sequence of number of vertices $\{n_l\}_{l \geq 1}$ for which the constructions work satisfies $n_{i+1} - n_i = o(n_i)$ for sufficiently large i. As a consequence, Spielman [175] proves that it is possible to get expander graphs of every size with only a moderate loss in expansion, and uses this fact in his constructions of expander codes. The same argument will also work for us. Alternatively, since the sequence of graphs is dense, we can pad each codeword of the left code with a small number of additional 0's so that its blocklength exactly matches the number of vertices in an explicit Ramanujan graph construction, and then apply our construction. This "padding" will affect the relative distance, rate and list decoding radius of the left code only by a negligible amount, and will essentially have no impact on any of the bounds we claim for the overall code construction. Therefore, in order to keep things simple, in our constructions of this chapter, as well as those in Chapter 11, we will ignore the above issue and simply assume that the blocklength of our "left code" and the number of vertices in the "expander" graph match exactly.

Theorem 9.20. *For all $\varepsilon > 0$, there exists a code family with the following properties:*

(i) (Rate and alphabet size) It has rate $\Omega(\varepsilon^2)$ and is defined over an alphabet of size $2^{O(\varepsilon^{-1} \log(1/\varepsilon))}$.

(ii) (Constructibility) A description of a code of blocklength N in the family can be constructed in deterministic $N^{O(\varepsilon^{-1})}$ time. A randomized Monte Carlo construction that has the list decodability claimed in (iii) with high probability can be obtained in probabilistic $O(\log^2 N\varepsilon^{-1}\log(1/\varepsilon))$ time.

(iii) (List decodability) A code of blocklength N in the family can be list decoded from up to $(1-\varepsilon)N$ errors in $O(N^2\varepsilon^{-O(1)}\log N)$ time using lists of size $O(1/\varepsilon)$.

Proof: The basic idea is to first construct a code C with good list recoverability properties by concatenating a Reed-Solomon code C_{RS} of rate $\Omega(\varepsilon)$ with a constant rate inner code C_{in} as guaranteed in Lemma 9.12. We will then apply the construction of Proposition 9.18 to obtain a code list decodable up to a fraction $(1-\varepsilon)$ of errors. Since the rate of the concatenated code is $\Theta(\varepsilon)$, and applying Proposition 9.18 incurs a further ε factor loss in the rate, we will get an overall rate of $\Omega(\varepsilon^2)$. The formal details follow. The basic structure of the construction is depicted in Figure 9.4.

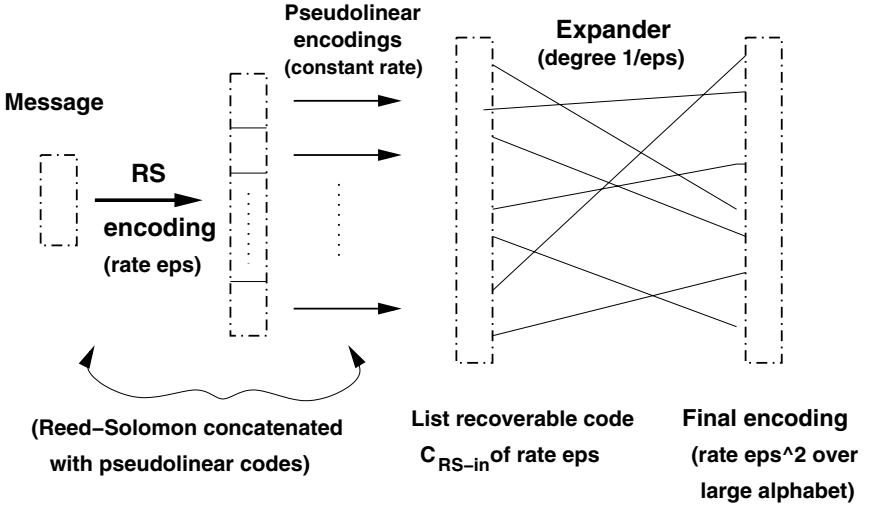

Fig. 9.4. Basic structure of code construction that achieves rate $\Omega(\varepsilon^2)$ and list decoding radius $(1-\varepsilon)$. The list recoverability of the concatenated code $C_{\mathrm{RS-in}}$, together with the expander, implies list decodability of the final code from a fraction $(1-\varepsilon)$ of errors.

Let m be any sufficiently large integer. Let $q_0 = O(1/\varepsilon^2)$ be a power of 2, and let F be a field of cardinality q_0^m. Let n_0 be in the range $q_0^{m-1} \le n_0 \le q_0^m$, $k_0 = \Theta(\varepsilon n_0)$, and C_{RS} be the Reed-Solomon code over F of blocklength n_0 and dimension k_0 (so C_{RS} has rate $\Theta(\varepsilon)$). Let C_{in} be a pseudolinear code

over $\mathbb{F}_{q_0}$ that maps m symbols over $\mathbb{F}_{q_0}$ (or, alternatively, a symbol of F) into $n_1 = O(m)$ symbols over $\mathbb{F}_{q_0}$, and further is $(1/4, O(1/\varepsilon), O(1/\varepsilon))$-list recoverable. Such a code C_{in} exists by Lemma 9.12.

Define $C_{\text{RS}-\text{in}}$ to be the code obtained by concatenating C_{RS} as outer code with C_{in} as inner code. $C_{\text{RS}-\text{in}}$ is a code of blocklength $N = n_0 \cdot n_1 = O(mq_0^m)$ and rate $\Omega(\varepsilon)$ over $\mathbb{F}_{q_0}$. The codewords in $C_{\text{RS}-\text{in}}$ can be divided into n_0 blocks of n_1 symbols each, corresponding to the encodings of the n_0 outer Reed-Solomon codeword symbols.

The final code C^* will be obtained from $C_{\text{RS}-\text{in}}$ using the construction of Proposition 9.18 (i.e., by redistributing the symbols of a codeword of $C_{\text{RS}-\text{in}}$ using an $(N, O(1/\varepsilon), 1/2, \varepsilon)$-disperser). It is readily checked that C^*, thus defined, is a code of blocklength N and rate $\Omega(\varepsilon^2)$ over an alphabet of size $q_0^{O(1/\varepsilon)} = 2^{O(\varepsilon^{-1}\log(1/\varepsilon))}$, proving Part (i) of the claim of the theorem.

The significant component in constructing C^* is finding the inner code C_{in} with the properties guaranteed in Lemma 9.12. Thus C^* can be constructed deterministically in $O(q_0^{O(m\varepsilon^{-1})}) = N^{O(\varepsilon^{-1})}$ time, or probabilistically in $O(m^2\varepsilon^{-1}\log q_0) = O(\log^2 N\varepsilon^{-1}\log(1/\varepsilon))$ time, as claimed in Part (ii) of the theorem statement.

It remains to prove the claim about the list decodability of C^*. For this, it suffices to prove that $C_{\text{RS}-\text{in}}$ is $(1/2, O(1/\varepsilon), O(1/\varepsilon))$-list recoverable in $O(N^2)$ time, since then the claim about list decoding C^* from a fraction $(1-\varepsilon)$ of errors will follow from the properties of C^* guaranteed by Proposition 9.18.

Suppose we are given lists L_i each consisting of at most $O(1/\varepsilon)$ elements of $\mathbb{F}_{q_0}$, for $1 \le i \le N$. We will present an $O(N^2\varepsilon^{-O(1)}\log N)$ time algorithm to find all codewords $\langle d_1, d_2, \ldots, d_N\rangle$ of $C_{\text{RS}-\text{in}}$ which satisfy $d_i \in L_i$ for at least $N/2$ values of i, $1 \le i \le N$. Recalling that a codeword in $C_{\text{RS}-\text{in}}$ comprises of n_0 blocks of n_1 symbols each, the lists L_i can be viewed as lists $L'_{j,s}$ for the possible symbols in position s of the codeword of C_{in} that encodes the j'th symbol of the Reed-Solomon codeword, for $1 \le s \le n_1$ and $1 \le j \le n_0$. Now consider the following list recovering procedure for $C_{\text{RS}-\text{in}}$. In the first step, the n_0 inner codes are decoded by brute-force by going over all codewords — namely, for each j, $1 \le j \le n_0$, one produces a list $\hat{L}_j$ of all elements of F whose encoding as per C_{in} contains an element from $L'_{j,s}$ for at least a fraction $1/4$ of the values of s. By the list recoverability property of C_{in} we have $|\hat{L}_j| = O(1/\varepsilon)$ for each j, $1 \le j \le n_0$. Note that all the inner decodings can be performed in $O(n_0^2/\varepsilon)$ time.

In the second step of the decoding, we run the list recovering algorithm for Reed-Solomon codes implied by the result of Theorem 6.21, to find a list $\mathcal{L}$ consisting of all messages $\mathbf{x}$ whose Reed-Solomon encoding contains an element of $\hat{L}_j$ for at least $n_0/4$ values of j, $1 \le j \le n_0$. Specifically, we apply the result of Theorem 6.21 with the choice $n = n_0$, $k = k_0 = O(\varepsilon n_0)$, $\ell \le \max_j |\hat{L}_j| = O(1/\varepsilon)$, and $\alpha = 1/4$ (one can check that the condition $\alpha > \sqrt{2k\ell/n}$ can be met for these values, with suitable constants in the big-

Oh notation). The decoding returns lists of size $O(\sqrt{\frac{n_0}{\varepsilon k_0}}) = O(1/\varepsilon)$, and can certainly be performed in $O(n_0^2 \varepsilon^{-O(1)} \log^3(q_0^m)) = O(n_0^2 \varepsilon^{-O(1)} \log^3 n_0)$ time. Since $n_0 = O(N/\log_{q_0} N)$, the time for list recovering the Reed-Solomon code is $O(N^2 \varepsilon^{-O(1)} \log N)$.

The final step prunes the list output by the Reed-Solomon decoder to include only those messages whose encodings as per $C_{\mathrm{RS-in}}$ contain an element of L_i for at least $N/2$ values of i, and then outputs this pruned list. The overall decoding time is dominated by the Reed-Solomon decoding time and is $O(N^2 \varepsilon^{-O(1)} \log N)$.

We now argue the correctness of the list recovering procedure. Let $\mathbf{x}$ be a message whose encoding $C^*(\mathbf{x}) = \langle d_1, d_2, \ldots, d_N \rangle$, where $d_i \in L_i$ for at least $N/2$ values of i. The codeword $\langle d_1, d_2, \ldots, d_N \rangle$ can also be viewed as consisting of symbols $b_{j,s}$ for $1 \leq j \leq n_0$ and $1 \leq s \leq n_1$, where $\langle b_{j,1}, b_{j,2}, \ldots, b_{j,s} \rangle$ is the block of the codeword corresponding to the inner encoding of the j'th symbol of $C_{\mathrm{RS}}(\mathbf{x})$. Let $J \subseteq [n_0]$ be the set of all j, $1 \leq j \leq n_0$, for which $b_{j,s}$ belongs to the corresponding list $L'_{j,s}$ for at least a fraction $1/4$ of values of s in the range $1 \leq s \leq n_1$. If $d_i \in L_i$ for at least $n_0 n_1/2$ values of i, by a simple averaging argument we get that $|J| \geq n_0/4$. Now, by the $(1/4, O(1/\varepsilon), O(1/\varepsilon))$-list recoverability property of C_{in}, for each $j \in J$, the list $\hat{L}_j$ contains the correct symbol of the Reed-Solomon encoding of the concerned message $\mathbf{x}$. Since $|J| \geq n_0/4$, the condition under which the Reed-Solomon list decoder outputs a message is satisfied by $\mathbf{x}$, and therefore it will output $\mathbf{x}$. Hence the message $\mathbf{x}$ will be included in the list output by the algorithm, as we desired to show. □

9.4.4 Better Rate with Sub-exponential Decoding

In the proof of Theorem 9.20, we used an outer Reed-Solomon code over a field of size linear in the blocklength. This implied that the dimension of the inner pseudolinear code was at most $\log N$, enabling a deterministic polynomial time algorithm to find the necessary pseudolinear code. We now indicate how at the cost of sub-exponential (about $2^{O(\sqrt{N})}$) construction and decoding time, we can improve the rate of the construction of Theorem 9.20 from $\Omega(\varepsilon^2)$ to $\Omega(\varepsilon)$, which is optimal up to constant factors. We will keep the discussion informal since in the next section we will generalize this result and state formal theorems anyway.

The idea is to perform the same construction as in Theorem 9.20, except we use Reed-Solomon codes of *constant* (independent of ε) rate, blocklength $\sqrt{n}$, over an alphabet of size $q_0^{\sqrt{n}}$ where $q_0 = O(1/\varepsilon^2)$. For the inner code, we use a constant rate $(1/4, O(1/\varepsilon), O(1/\varepsilon))$-list recoverable pseudolinear code of dimension $\sqrt{n}$ (i.e., same as in Theorem 9.20, except with larger dimension). Note that the concatenated code also has constant rate, and the dominant component in its construction is once again the pseudolinear code construction, which takes $2^{O_\varepsilon(\sqrt{n})}$ time to perform deterministically, and $O_\varepsilon(n)$ time

to perform probabilistically (here by the O_ε notation we are hiding also constant factors that depend on ε). We claim that the concatenated code can be $(1/2, O(1/\varepsilon), L)$-list recovered in $2^{O_\varepsilon(\sqrt{n})}$ time (for $L = 2^{O_\varepsilon(\sqrt{n})}$). At the first step, all the inner codes are $(1/4, O(1/\varepsilon), O(1/\varepsilon))$-list recovered by a brute-force search over all codewords in $q_0^{O(\sqrt{n})}$ time. This passes lists of size $O(1/\varepsilon)$ for the possible symbol at each position of the Reed-Solomon codeword. The decoding is completed by going over every set of a fraction $1/4$ of these lists and every choice of symbols from each of these lists, and for each of them checking if there is a Reed-Solomon codeword consistent with those symbols. This brute-force procedure takes about $(1/\varepsilon)^{O(\sqrt{n})}$ time and succeeds in $(1/4, O(1/\varepsilon), L)$-list recovering the Reed-Solomon code. The correctness of this procedure follows using arguments similar to those of Theorem 9.20.

We thus have a constant rate code which is $(1/2, O(1/\varepsilon), L)$-list recoverable in $2^{O_\varepsilon(\sqrt{n})}$ time. Using this in the construction of Proposition 9.18, we can get a rate $\Omega(\varepsilon)$ code C^* list decodable in $2^{O_\varepsilon(\sqrt{n})}$ time from up to a fraction $(1 - \varepsilon)$ of errors, as we desired to show.

To get binary codes, we can concatenate C^* with a binary code of rate $\Omega(\varepsilon^2)$ which has list decoding radius $(1/2 - O(\varepsilon))$ for a list size of $O(1/\varepsilon^2)$. Such codes exist by the result of Theorem 5.8 (from Chapter 5). This gives binary codes of rate $\Omega(\varepsilon^3)$ that are list decodable in $2^{O_\varepsilon(\sqrt{n})}$ time from up to a fraction $(1/2 - \varepsilon)$ of errors. Note that the rate is better than the result of Theorem 8.11 that achieved a rate of $\Omega(\varepsilon^4)$. However, the construction and decoding time are no longer polynomial in the blocklength.

In the next section, we present a more complicated scheme to improve the decoding time to $2^{O(N^\gamma)}$ for any desired $\gamma > 0$. The spirit of the construction is the same as in this section; the details are however more complicated.

9.5 Constructions with Better Rate Using Multi-concatenated Codes

We now introduce a code construction where an outer Reed-Solomon code is concatenated with multiple levels of inner codes (as guaranteed by Lemma 9.6, albeit over large, growing sized alphabets which decrease in size from the outermost to innermost levels). We call such codes *multi-concatenated codes*, which are discussed in Section 9.5.1. We will then, in Section 9.5.2, use these codes to prove Theorem 9.22 which allows us improve the rate (from Theorem 9.20) by an ε factor at the expense of the decoding time becoming sub-exponential in the blocklength. This gives our construction (2a), and yields codes of the optimal $\Omega(\varepsilon)$ rate that have list decoding algorithms of "reasonable" complexity for correcting a fraction $(1 - \varepsilon)$ of errors. Following this, in Section 9.5.3, we will concatenate these codes with appropriate binary codes to get our construction (2b), i.e., binary codes of rate $\Omega(\varepsilon^3)$ list decodable in sub-exponential time from up to a fraction $(1/2 - \varepsilon)$ of errors.

9.5.1 The Basic Multi-concatenated Code

We now describe the construction of multi-concatenated codes and their properties. This is stated formally in the lemma below. The result is similar to Lemma 9.12 in terms of the parameters of the codes it guarantees. In fact, for the case $p = 1$, the result is in fact just that of Lemma 9.12 (with the claimed decoding time being that of the naive decoding algorithm that does a brute-force search over all possible codewords).

For larger values of p, the construction is somewhat messy. The result for larger values of p is necessary only to improve the decoding time from the $2^{O(\sqrt{N})}$ bound that was presented in Section 9.4.4 to $2^{O(N^\gamma)}$. The reader might want to take the result of the lemma below as a black-box in the first reading and come back to its proof if interested after seeing its applications in Sections 9.5.2 and 9.5.3 (the case $p = 1$ for those applications gives precisely the constructions outlined in Section 9.4.4).

Lemma 9.21. *For every $p \geq 1$ and all sufficiently small $\varepsilon > 0$, there exist a code family with the following properties:*

(i) (Rate and alphabet size) The family has rate $2^{-O(p^2)}$ and is defined over an alphabet of size $O(1/\varepsilon^2)$.

(ii) (List decodability property) Each member of the code family is $(\frac{1}{2}, \frac{1}{\varepsilon}, \frac{2^{O(p^2)}}{\varepsilon})$-list recoverable. Furthermore such list decoding can be accomplished in $2^{O(N^{1/p}\log(1/\varepsilon))}$ time, where N is the blocklength of the concerned code.

(iii) (Constructibility) A code of blocklength N in the family can be constructed in probabilistic $O(N^2 \log(1/\varepsilon))$ time (the code will have the list decodability claimed in (ii) with high probability). A deterministic construction can be obtained in $2^{O(N\varepsilon^{-1}\log(1/\varepsilon))}$ time. Also, encoding can be performed in $O(N^2 \log^{O(1)}(1/\varepsilon))$ time.

Proof Idea. The basic idea is to use p levels of large alphabet pseudolinear codes as guaranteed by Lemma 9.15 in a suitable concatenation scheme. These codes will be defined over progressively decreasing alphabet size. The outermost code will be a constant-rate code of blocklength $O(N^{1/p})$ defined over an alphabet of size $q^{2\cdot N^{(p-1)/p}}$ (where $q = O(1/\varepsilon^2)$). Each symbol of this codeword will then be encoded by another code guaranteed by Lemma 9.15, this time over a smaller alphabet $\mathrm{GF}(q^{2\cdot N^{(p-2)/p}})$, but again of blocklength $O(N^{1/p})$ and constant rate. Each symbol of this encoding will be further encoded by a similar constant rate code of blocklength $O(N^{1/p})$, but over an even smaller alphabet $\mathrm{GF}(q^{2\cdot N^{(p-3)/p}})$, and so on. This will continue for several more levels till the alphabet size is down to $\mathrm{GF}(q^{2\cdot N^{1/p}})$. Finally each of the field symbols is encoded by one final constant rate pseudolinear code over $\mathrm{GF}(q)$ of dimension $2N^{1/p}$.

The big plus of using p levels is that the code at each level has dimension and blocklength $O(N^{1/p})$. Since the decoding time guaranteed by Lemma 9.15 was about $(1/\varepsilon)^{O(m)}$ where m was the dimension, we can exploit the "fast" decoding of the codes at each level to give a decoding algorithm for the overall multi-concatenated code with runtime exponential in $N^{1/p}$, or sub-exponential in N.

All in all, given a list L_i of $O(1/\varepsilon)$ symbols of GF(q) for each of the N positions of the final codeword, the successive decodings pass up a list of $O(1/\varepsilon)$ symbols (with larger and larger constants in the big-Oh notation) for each position of each pseudolinear codeword. Finally, after p levels of decoding, we will recover a list of at most $O(1/\varepsilon)$ codewords which includes all codewords that agree with an element of L_i for at least $N/2$ values of i.

The formal proof given below just follows the above idea, though it necessarily involves a somewhat careful choice of parameters to ensure that the decodings all work together to give the claimed list recoverability property. The reader satisfied with the above proof idea should feel free to skip it.

Proof: Let q be a power of 2 with $q = O(1/\varepsilon^2)$ – we will define a code over $\mathbb{F}_q$. We will describe the code family by describing a code C_p that encodes $\mathbf{x} \in \mathbb{F}_q^n$ into $C_p(\mathbf{x}) \in \mathbb{F}_q^{n \cdot 2^{O(p^2)}}$, for any large enough n which is of the form $n = 2 \cdot m^p$ for some integer m. The code C_p is described below inductively for increasing values of p.

Code Description. For $p = 1$, the code C_1 will be a q-ary $(\alpha_1, 1/\varepsilon, c/\varepsilon)$-list recoverable code that encodes a string of length $2m$ over $\mathbb{F}_q$ into a codeword of length $a_1 m$ (for suitable constants $a_1, c > 1$ and $\alpha_1 < 1$). Such a code is guaranteed to exist by Lemma 9.12.

For $p > 1$, the code C_p will be a q-ary code of dimension $2m^p$. We defined the code C_p inductively using C_{p-1} and a p'th level code G_p defined as follows. G_p will be a code defined over an alphabet Σ_p of size $q^{2m^{p-1}}$ as guaranteed by Lemma 9.15 (using the choice $s = m^{p-1}$ in that lemma). Specifically, G_p has dimension m and blocklength $a_p m$, where a_p is a constant that depends only on p but is independent of ε. Moreover, G_p is $(\alpha_p, c^{p-1}/\varepsilon, c^p/\varepsilon)$-list recoverable, for a suitable constant $\alpha_p > 1$ (the details on how to pick the constants will be clarified shortly).

We now give an inductive definition of C_p in terms of C_{p-1} and the above code G_p. To encode $\mathbf{x} \in \mathbb{F}_q^n$ using the code C_p, where $n = 2m^p$, we view $\mathbf{x}$ as a string of length m over GF($q^{2m^{p-1}}$) , and first encode it using G_p. This gives us a string $\mathbf{x_1}$ of $a_p m$ symbols over GF($q^{2m^{p-1}}$). We now view each of these $a_p m$ symbols as a string of length $2m^{p-1}$ over $\mathbb{F}_q$ and independently encode them using C_{p-1}. This completes the inductive specification of the code C_p.

The list recoverability requirement on C_p will let us fix the constants α_j's above. This will in turn fix the rates of the codes (or in other words the

constant a_j's). We sketch this next, followed by an analysis of the construction complexity (both probabilistic and deterministic) of the code C_p.

Rate of the Construction. For every $p \geq 1$, and each fixed $\alpha < 1$, for a large enough constant $c = c_{p,\alpha}$, we now wish to pick parameters (specifically α_j's) that allow us to show that the code C_p constructed above is $(\alpha, 1/\varepsilon, c^p/\varepsilon)$-list recoverable in $2^{O(N^{1/p}\log(1/\varepsilon))}$ time where N is the blocklength of C_p. This can be achieved for $p = 1$ by a choice of C_1 with $\alpha_1 = \alpha$ and the rate of the code is an absolute constant (that depends on α). For $p > 1$, let us by induction pick C_{p-1} so that it is $(\alpha/2, 1/\varepsilon, c^{p-1}/\varepsilon)$-list recoverable. We will pick the "outermost" code G_p in the construction of C_p so that it is $(\alpha/2, c^{p-1}/\varepsilon, c^p/\varepsilon)$-list recoverable. By Lemma 9.15 we have such a G_p with rate

$$R(G_p) \geq \frac{1}{6}(\alpha/2 - 1/c) \; . \tag{9.6}$$

Now, applying a standard averaging argument one can combine the facts that C_{p-1} is $(\alpha/2, 1/\varepsilon, c^{p-1}/\varepsilon)$-list recoverable and G_p is $(\alpha/2, c^{p-1}/\varepsilon, c^p/\varepsilon)$-list recoverable to conclude that C_p is $(\alpha, 1/\varepsilon, c^p/\varepsilon)$-list recoverable. It remains to estimate the rate $R(C_p)$ of the code C_p (as a function f of α, p). By the above construction, we have

$$f(\alpha, p) = R(G_p) f(\alpha/2, p-1) \geq \frac{1}{6}\Big(\frac{\alpha}{2} - \frac{1}{c}\Big) f(\alpha/2, p-1) \qquad \text{(using (9.6))} \; .$$

Unwinding the recurrence, for α a fixed constant, like $\alpha = 1/2$ say, we can get C_p that is $(1/2, 1/\varepsilon, c^p/\varepsilon)$-list recoverable with $c \simeq O(2^p)$ and rate $R(C_p) = 2^{-O(p^2)}$. We have thus verified Property (i) for our code construction.

Decoding Complexity. The decoding of the code C_p proceeds inductively from the lowermost levels of the concatenation upwards. This is also best described inductively. For $p = 1$, as mentioned earlier, the decoding of C_1 proceeds by running over all $q^{O(m)} = q^{O(N)}$ codewords. For $p > 1$, given lists of size $1/\varepsilon$ at each position of the code, each of $O(m)$ codes C_{p-1} used to encode the symbols of G_p can be list recovered by induction in $2^{O(m\log(1/\varepsilon))}$ time. This passes a list of c^{p-1}/ε possible symbols for each of the $a_p m$ positions of the code G_p. The code G_p is then list recovered to produce a final set of c^p/ε messages as the answers. Since G_p is picked as guaranteed by Lemma 9.15, the list recovering of G_p can be performed in $(c^{p-1}/\varepsilon)^{O(m)} = 2^{O(m\log(1/\varepsilon))}$ time as well (absorbing factors which depend on c^{p-1} into the big-Oh notation, since we treat c, p as fixed constants). The overall decoding time is the sum of the decoding times for C_{p-1} and G_p, and is thus $2^{O(m\log(1/\varepsilon))}$. Since $n = 2m^p$ is the length of the message and the rate of C_p is $2^{-O(p^2)}$, we have the overall blocklength $N = 2^{O(p^2)} n = O(m^p)$. Therefore, the overall decoding complexity equals $2^{O(N^{1/p}\log(1/\varepsilon))}$, as claimed in Part (ii) of the lemma.

Construction Complexity. We finally verify the claimed construction complexity bounds for the code C_p. For $p = 1$, we appeal to Lemma 9.12 to conclude that C_1 can be constructed in $O(N^2\varepsilon^{-1}\log(1/\varepsilon))$ probabilistic time, or $2^{O(N\varepsilon^{-1}\log(1/\varepsilon))}$ deterministic time. For $p > 1$, the dominant component is the time to construct the outermost code G_p. Lemma 9.15 implies that G_p can be constructed in E_p can be constructed in $O(m^{2p}\log(1/\varepsilon))$ time probabilistically, and in $2^{O(m^p\varepsilon^{-1}\log(1/\varepsilon))}$ time deterministically. Since $m = O(N^{1/p})$, the construction time is $O(N^2\log(1/\varepsilon))$ probabilistically, and $2^{O(N\varepsilon^{-1}\log(1/\varepsilon))}$ deterministically. The encoding time is again dominated by the time to perform the outermost encoding according to G_p, and is therefore $O(m^{2p}\log^2 q) = O(N^2\log^2(1/\varepsilon))$. This completes the proof of Property (iii) in the statement of the lemma. □

9.5.2 Codes of Rate $\Omega(\varepsilon)$ with Sub-exponential List Decoding for a Fraction $(1-\varepsilon)$ of Errors

We now use the multi-concatenated codes from the previous section to attain rate $\Omega(\varepsilon)$ for codes list decodable up to a fraction $(1-\varepsilon)$ of errors in sub-exponential time. Note that such a result was also discussed in Section 9.4.4, but we will now improve the decoding time from $2^{O(\sqrt{N})}$ to $2^{O(N^\gamma)}$ for each fixed $\gamma > 0$. Setting $\gamma = 1/2$ in the below theorem gives the result claimed in Section 9.4.4.

Theorem 9.22. *For every constant $\gamma > 0$ the following holds: for all sufficiently small $\varepsilon > 0$, there exists a code family with the following properties:*

- *(i) (Rate and alphabet size) The code has rate $\Omega(\varepsilon 2^{-O(\gamma^{-2})})$ and is defined over an alphabet of size $2^{O(\varepsilon^{-1}\log(1/\varepsilon))}$.*
- *(ii) (Construction complexity) A description of a code of blocklength N in the family can be constructed in probabilistic $O(N^{2-2\gamma}\log(1/\varepsilon))$ time, or deterministically in time $2^{O(N^{1-\gamma}\varepsilon^{-1}\log(1/\varepsilon))}$. Moreover the code can be encoded in $O(N^{2(1-\gamma)}\log^2 N\log^{O(1)}(1/\varepsilon))$ time.*
- *(iii) (List decodability) The code can be list decoded in $2^{O(N^\gamma\log(1/\varepsilon))}$ time from up to a fraction $(1-\varepsilon)$ of errors.*

Proof: We use a construction quite similar to that of Theorem 9.20. Let $p' = \lceil 1/\gamma \rceil$ and $q_0 = O(1/\varepsilon^2)$ be a prime power. At the outermost level, we use a Reed-Solomon code C_{RS} of blocklength n_0 over a field of size $q_0^{n_0^{p'-1}}$ (instead of a field of size n_0 that we used in earlier constructions). Furthermore, the rate of the Reed-Solomon code will now be an absolute constant, say $1/4$ (as opposed to $O(\varepsilon)$ earlier). Each of the n_0 field elements (viewed as a string of length $n_0^{p'-1}$ over $\mathrm{GF}(q_0)$) is encoded using a *multi-concatenated* inner code C'_{in} that encodes $n_0^{p'-1}$ symbols into $2^{O(p^2)}n_0^{p'-1}$ symbols (over $\mathrm{GF}(q_0)$) and which has the properties guaranteed by Lemma 9.21 for $p = p' - 1$. Denote

by $C_{\mathrm{RS-in}}$ the resulting concatenated code. The rest of the construction (i.e. obtaining the final code C^* from $C_{\mathrm{RS-in}}$ using a disperser) is the same as Theorem 9.20, and the claims about the rate and alphabet size follow similarly to Theorem 9.20. See Figure 9.5 for a sketch of the basic components in the construction.

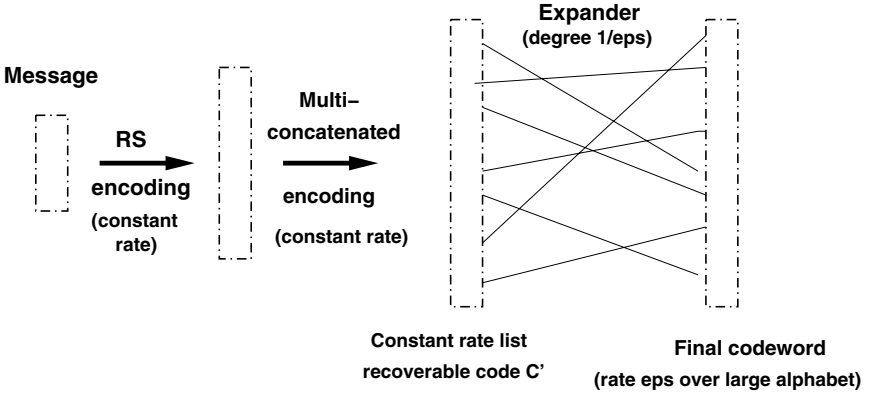

Fig. 9.5. Basic structure of code construction that achieves rate $\Omega(\varepsilon)$ and list decoding radius $(1-\varepsilon)$. The list recoverability property of C' enables list decoding of the final code from a fraction $(1-\varepsilon)$ of errors.

About construction complexity, the significant component is finding the inner code C'_{in}, which can be done in $2^{O(n_0^{p'-1}\varepsilon^{-1}\log(1/\varepsilon))}$ time by Lemma 9.21, or in probabilistic $O\big((n_0^{p'-1})^2\log(1/\varepsilon)\big)$ time. Since the overall blocklength of the code equals $N = n_0 2^{O(p^2)} n_0^{p'-1} = 2^{O(p^2)} n_0^{p'}$, we have $n_0 = O(N^{1/p'})$ and hence the claimed bounds on the construction time follow.

About list decoding complexity, we claim that $C_{\mathrm{RS-in}}$ is $(1/2, O(1/\varepsilon), L)$-list recoverable in $2^{O(N^{1/p'}\log(1/\varepsilon))}$ time, for $L = 2^{O(N^{1/p'}\log(1/\varepsilon))}$. Now, appealing to Proposition 9.18 implies that our final code C^* will then be $((1-\varepsilon)N, L)$-list decodable in $2^{O(N^{1/p'}\log(1/\varepsilon))}$ time, which is what we would like to show.

To $(1/2, O(1/\varepsilon), L)$-list recover the concatenated code $C_{\mathrm{RS-in}}$, we first use the decoding strategy guaranteed by Lemma 9.21 to $(1/4, O(1/\varepsilon), O(1/\varepsilon))$-list recover each of the n_0 inner codes. This takes a total of $n_0 2^{O((n_0^{p'-1})^{1/p}\log(1/\varepsilon))} = 2^{O(n_0\log(1/\varepsilon))}$ time (since $p = p' - 1$), and passes lists of size $O(2^{O(p^2)}/\varepsilon)$ corresponding to each position of the Reed-Solomon code. Since we are thinking of p as a constant and ε as sufficiently small, we can assume that lists of size $O(1/\varepsilon)$ are passed for each position of the

Reed-Solomon code. For any message $\mathbf{x}$ that is a solution to the list recovering instance, at least a fraction $1/4$ of these lists contain the "correct" symbol of $C_{\mathrm{RS}}(\mathbf{x})$. We now finish the decoding by a brute-force decoding of the outermost Reed-Solomon code as follows. Given lists of size $O(1/\varepsilon)$ for each of the n_0 codeword positions (these lists are the ones obtained after the independent decoding of the n_0 inner codes), for each subset of $n_0/4$ codeword positions and each possible choice of field element from the respective list (this involves considering $\binom{n_0}{n_0/4} \cdot (O(1/\varepsilon))^{n_0/4} = 2^{O(n_0 \log(1/\varepsilon))}$ possibilities), we do the following. Determine if there is a Reed-Solomon codeword consistent with the $n_0/4$ field elements at the chosen positions (this can be performed using a straightforward polynomial interpolation since the rate of the Reed-Solomon code is $1/4$), and, if so, include that codeword in the list. Recalling that $n_0 = O(N^{1/p'})$, it is clear that the Reed-Solomon decoding can be performed in $2^{O(N^{1/p'} \log(1/\varepsilon))}$ time. Since $1/p' \leq \gamma$, this is consistent with our claimed runtime. □

Improvement to List Size Note that the size of the list returned in decoding the above codes up to a fraction $(1-\varepsilon)$ of errors is $2^{O(N^{\gamma} \log(1/\varepsilon))}$. It might be of interest to keep this list size small, ideally a constant, even if the decoding algorithm itself runs in sub-exponential time. This can be achieved by skipping the use of the outermost Reed-Solomon code in the above construction and just using the $(1/2, O(1/\varepsilon), O(1/\varepsilon))$-list recoverable multi-concatenated code C'_{in} in the construction of Proposition 9.18. This will also give a code that is $((1-\varepsilon)N, O(1/\varepsilon))$-list decodable in time $2^{O(N^{\gamma} \log(1/\varepsilon))}$, at the expense of the construction times being slightly worse than those claimed in Theorem 9.22. Specifically, the probabilistic construction time will now be $O(N^2 \log(1/\varepsilon))$ and the deterministic construction time will be $2^{O(N\varepsilon^{-1} \log(1/\varepsilon))}$.

A Version of Theorem 9.22 for List Recoverability We now state a variant of Theorem 9.22 that will be useful in getting binary codes in the next section.

Lemma 9.23. *For every constant $\gamma > 0$ the following holds: for all $\varepsilon > 0$, there exists a code family with the following properties:*

(i) (Rate and alphabet size) The code has rate $\Omega(\varepsilon 2^{-O(\gamma^{-2})})$ and is defined over an alphabet of size $2^{O(\varepsilon^{-1} \log(1/\varepsilon))}$.

(ii) (Construction complexity) A description of a code of blocklength N in the family can be constructed in probabilistic $O(N^{2-2\gamma} \log(1/\varepsilon))$ time, or deterministically in time $2^{O(N^{1-\gamma}\varepsilon^{-3} \log(1/\varepsilon))}$.

(iii) (List decodability) The code can be $(\varepsilon/2, O(1/\varepsilon^2), L)$-list recovered in $2^{O(N^{\gamma} \log(1/\varepsilon))}$ time (for $L = 2^{O(N^{\gamma} \log(1/\varepsilon))}$).

Proof (Sketch): The above result really follows using the same proof as that of Theorem 9.22. The point is that we we assume the code $C_{\mathrm{RS-in}}$ to

be $(1/2, O(1/\varepsilon^3), L)$-list recoverable (instead of $(1/2, O(1/\varepsilon), L)$-list recoverable). Accordingly we will have to change its parameters and replace each ε by ε^3. But this will still keep its alphabet size $q_0 = 1/\varepsilon^{O(1)}$ and its rate will be $2^{-O(\gamma^{-2})}$ which is a constant independent of ε. We will get our final code C^* from the code $C_{\mathrm{RS-in}}$ by applying Lemma 9.19 (instead of Proposition 9.18), with the choice $\ell = O(1/\varepsilon^2)$. Thus we can get a code C^* of rate $\Omega(\varepsilon)$ over an alphabet of size $q_0^{O(1/\varepsilon)} = 2^{O(\varepsilon^{-1}\log(1/\varepsilon))}$ which is $(\varepsilon/2, O(1/\varepsilon^2), L)$-list recoverable. □

9.5.3 Binary Codes of Rate $\Omega(\varepsilon^3)$ with Sub-exponential List Decoding Up to a Fraction $(1/2 - \varepsilon)$ of Errors

We now use the code construction from Lemma 9.23 as outer codes in a concatenated scheme with a suitable binary inner code and obtain constructions of good list decodable binary codes. Our result is stated formally below.

Theorem 9.24. *For every constant $\gamma > 0$ the following holds: for all sufficiently small $\varepsilon > 0$, there exists a* binary *code family with the following properties:*

(i) (Rate) It has rate $\Omega(\varepsilon^3 2^{-O(\gamma^{-2})})$.

(ii) (Construction Complexity) A description of a code of blocklength N from the family can be constructed with high probability in randomized $O\big((N^{2(1-\gamma)} + \varepsilon^{-6})\log(1/\varepsilon)\big)$ time or deterministically in time $2^{O(N^{1-\gamma}\varepsilon^{-3}\log(1/\varepsilon))}$. The code can be encoded in $O(N^{2(1-\gamma)}\log^2 N \log^{O(1)}(1/\varepsilon))$ time.

(iii) (List decodability) A code of blocklength N from the family can be list decoded from up to $(1/2 - \varepsilon)N$ errors in $2^{O(N^{\gamma}\log(1/\varepsilon))}$ time.

Proof: The code will be obtained by concatenation of an outer code C_{out} over a large alphabet with a binary code C_{inner}. The code C_{out} will be picked as guaranteed by Lemma 9.23 and will be over an alphabet Σ_{out} of size $2^{O(\varepsilon^{-1}\log(1/\varepsilon))}$. Let m denote the blocklength of C_{out}. The code C_{inner} will be a binary code of rate $\Omega(\varepsilon^2)$, dimension $\lg|\Sigma_{\mathrm{out}}| = O(\varepsilon^{-1}\log(1/\varepsilon))$, blocklength $t = O(\varepsilon^{-3}\log(1/\varepsilon))$ which is $((\frac{1}{2} - \frac{\varepsilon}{2})t, O(1/\varepsilon^2))$-list decodable. Such codes C_{inner} (in fact, linear codes) exist and can be found deterministically in $2^{O(t)}$ time (cf. Section 5.3.2 and Section 8.6.1). Alternatively, one can also pick a random rate $\Omega(\varepsilon^2)$ pseudolinear code by investing $O(\varepsilon^{-6}\log^2(1/\varepsilon))$ randomness. The fact that this such a code will be $((1/2 - \varepsilon)t, O(1/\varepsilon^2))$-list decodable with high probability can be seen using Lemma 9.8 with the choice $q = 2$, $p = (1 - \varepsilon)/2$ and $L = O(1/\varepsilon^2)$.

Let us call the entire concatenated binary code C_{bin} and let its blocklength be $N = m \cdot t$. A codeword in C_{bin} is comprised of m blocks (of t bits each) corresponding to the m codeword positions of the outer code C_{out}. Note that C_{bin} clearly has the claimed rate since C_{out} has rate $\Omega(\varepsilon \cdot 2^{-O(\gamma^{-2})})$

and C_{inner} has rate $\Omega(\varepsilon^2)$. The construction complexity of C_{bin} is the time required to construct C_{out} plus that required to construct the binary code C_{inner}. Therefore, using Lemma 9.23 and the above discussion concerning the construction of C_{inner}, the claimed bound on the construction complexity of C_{bin} follows. This proves Properties (i) and (ii) claimed in the theorem.

It remains to prove Property (iii) concerning the list decodability of C_{bin}. By the list decodability property of C_{out} guaranteed by Lemma 9.23, we may assume that there is an efficient algorithm A_{out} with runtime exponential in m^{γ} that, given as input lists L_i of size $O(1/\varepsilon^2)$ for $1 \leq i \leq m$, can find a list of all codewords of $\langle c_1, \ldots, c_m \rangle \in C_{\text{out}}$ such that $c_i \in L_i$ for at least a fraction $\varepsilon/2$ of the i's.

The list decoding algorithm for C_{bin} works as follows. Given a received word $\mathbf{r} \in \{0,1\}^N$, the algorithm finds, for each i, $1 \leq i \leq m$, a list L_i of all symbols β of Σ_{out} such that $C_{\text{inner}}(\beta)$ differs from the i'th block $\mathbf{r_i}$ of $\mathbf{r}$ in at most $\frac{t(1-\varepsilon)}{2}$ positions. Since C_{inner} is $((1-\varepsilon)t/2, O(1/\varepsilon^2))$-list decodable, each L_i has at most $O(1/\varepsilon^2)$ elements. Now we run the decoding algorithm A_{out} with input these m lists L_i. The runtime of the algorithm is dominated by that of A_{out}, which is $2^{O(m^{\gamma}\log(1/\varepsilon))}$, and is thus within the claimed bound.

To prove correctness of the algorithm, let $\mathbf{c} \in C_{\text{bin}}$ be any codeword which differs from $\mathbf{r}$ in at most $(1/2 - \varepsilon)N$ positions (the list decoding algorithm must output every such $\mathbf{c}$). Let β_i, $1 \leq i \leq m$, be the i'th symbol of the codeword of C_{out} which upon concatenation with C_{inner} gives $\mathbf{c}$. By a simple averaging argument, one can show that for at least an $\varepsilon m/2$ values of i, $1 \leq i \leq m$, $\beta_i \in L_i$. By its claimed property, the decoding algorithm A_{out} will hence place $\mathbf{c}$ on the list it outputs. This completes the proof of Property (iii) claimed in the theorem as well. □

9.6 Improving the Alphabet Size: Juxtaposed Codes

One drawback of the result of Theorem 9.20 (as well as that of Theorem 9.22) is that these give codes over an alphabet which is exponentially large in $1/\varepsilon$. In this section, we indicate how one can improve the alphabet size significantly at the expense of a moderate worsening of the rate, by using an entirely different technique (the technique is also somewhat simpler, as it avoids the use of expanders). The basic idea is use to several concatenated codes, each one of which is "good" for some error distribution, and then "juxtapose" symbols from these codes together to obtain a new code over a larger alphabet which has nice list decodability properties. The idea of juxtaposed codes is already used in this chapter in the proof of Lemma 9.21, where we juxtaposed a pseudolinear code with a linear code. But the use of juxtaposition there was for a largely "technical" reason. On the other hand, juxtaposition is fairly natural for the codes we construct in this section. The discussion in Section 9.2.2 already presented a high level discussion of the rationale behind juxtaposed codes; we further elaborate on this aspect below.

9.6.1 Intuition

The basic intuition for considering juxtaposed codes can be understood by considering the following very natural way of constructing a code list decodable up to a fraction $(1 - \varepsilon)$ of errors. Namely, concatenate an outer Reed-Solomon code (call its blocklength n_0) with an inner code over an alphabet of size $O(1/\varepsilon^2)$ as guaranteed by Corollary 9.9. Each inner encoding by itself can tolerate a fraction $(1 - O(\varepsilon))$ of errors via list decoding with lists of size $O(1/\varepsilon)$. Now consider a received word $\mathbf{r}$ and a codeword $\mathbf{c}$ of the concatenated code which agree on a fraction ε of symbols. If this agreement is evenly distributed among the n_0 blocks that correspond to the various inner encodings, then each of the n_0 inner codes can be decoded (by a simple brute-force search over all inner codewords) and return a list of $O(1/\varepsilon)$ Reed-Solomon symbols that includes the "correct" symbol. If the rate of the Reed-Solomon code is $O(\varepsilon)$, list recovering the Reed-Solomon using these lists is guaranteed to include the codeword $\mathbf{c}$ (cf. Chapter 6). The overall rate of the concatenated code can thus be $\Omega(\varepsilon^2)$, since both the Reed-Solomon and inner codes can have rate $\Omega(\varepsilon)$.

This seems to give us the desired construction with rate $\Omega(\varepsilon^2)$. There is a (big) problem, however. There is no guarantee that errors will be evenly distributed among the n_0 blocks. In fact, on the other extreme, it is possible that $\mathbf{c}$ and $\mathbf{r}$ agree completely on a fraction ε of the blocks, and differ completely on the remaining fraction $(1 - \varepsilon)$ of the blocks. To tackle this case, the natural inner decoding to perform is to, for each block, simply return the symbol whose inner encoding is closest to that block of $\mathbf{r}$. Now the "correct" symbol (corresponding to $\mathbf{c}$) will be thus passed to the outer Reed-Solomon decoder for a fraction ε of the positions. To finish the decoding, we would need to be able to list decode the Reed-Solomon code for a $(1-\varepsilon)$ errors, and it is only known how to do so efficiently if the rate is $O(\varepsilon^2)$ (cf. Chapter 6, Theorem 6.16).

Thus the two widely differing (i.e. completely uniform and highly non-uniform) distributions of errors between the various inner codeword blocks require the rate of the Reed-Solomon code to be $O(\varepsilon)$ and $O(\varepsilon^2)$ respectively. Thus one has to conservatively pick the rate of the Reed-Solomon code to be $O(\varepsilon^2)$ to handle the highly non-uniform distribution of errors. The rate of the inner code has to be $O(\varepsilon)$ to handle the uniform distribution of errors. Therefore the overall rate has to be at most $O(\varepsilon^3)$.

A closer inspection of the question reveals that this limitation is due to our using a single outer code and single inner code, which can only be optimized for one error distribution, and suffers for a different error distribution. This suggests the use of several concatenated codes *in parallel*, each with its own outer and inner code rates that are optimized for a certain distribution of errors between the various inner codeword blocks. These concatenated codes can then be "put together" by juxtaposing their symbols together. Now, depending on how uniformly the errors are distributed, a certain concatenated

code "kicks in" and enables recovery of the message. The use of multiple concatenated codes reduces the rate compared to the expander based constructions, and also increases the alphabet size compared to a single concatenated code. It turns out, however, that we can still do much better on alphabet size than the bound of $2^{O(\varepsilon^{-1}\log(1/\varepsilon))}$ that was achieved by the construction of Theorem 9.20.

9.6.2 The Actual Construction

We first discuss the basic code construction scheme, and then formally state the theorems we obtain for appropriate setting of parameters. Let $\varepsilon > 0$ be given; the goal being to construct a code family of good rate (as close to $\Omega(\varepsilon^2)$ as possible) that can be efficiently decoded from up to a fraction $(1-\varepsilon)$ of errors. Let $t \geq 1$ be an integer parameter (t will be the number of codes that will be juxtaposed together).

Let $\delta_0, \delta_1, \ldots, \delta_t$ be a sequence in geometric progression with $\delta_0 = \varepsilon/2$, $\delta_t = 1$, and $\delta_i/\delta_{i-1} = \Delta$ for $1 \leq i \leq t$. Note that these parameters must satisfy $\Delta^t = 2/\varepsilon$.

Fix $c > 1$ and let $q_0 = O(1/\varepsilon^c)$ be a prime power. Let m be a large enough integer. The juxtaposed code construction, say C^*, that we now give, will be parameterized by $(q_0, m, \varepsilon, t, \Delta)$.

For each i, $1 \leq i \leq t$, we will have one q_0-ary concatenated code $\mathbf{C_i}$ with outer code a Reed-Solomon code C_i^{RS} and inner code an appropriate q_0-ary pseudolinear code C_i^{in}. The parameters of these codes will be as follows.

The Reed-Solomon Codes. The blocklength of each of the Reed-Solomon codes will be the same, $n_0 = q_0^m$. The code C_i^{RS} will be defined over the alphabet $\mathrm{GF}(q^{m_i})$ where $m_i = m\delta_i/\delta_0$. The rate of C_i^{RS} will be $R_i = \Theta(\varepsilon^2/(t^2\delta_i\Delta))$ and its dimension will be $k_i = R_i n_0$ (the reason for this choice of rate will be become clear once we specify the decoding algorithm in the proof of Theorem 9.25 below). Note that each message that is encoded by C_i^{RS} consists of k_i symbols over $\mathrm{GF}(q_0^{m_i})$, or equivalently, $k_i m_i = R_i n_0 m_i = \Theta(\frac{\varepsilon m n_0}{t^2\Delta})$ symbols over $\mathrm{GF}(q_0)$. This quantity is independent of i, and hence the number of q_0-ary symbols in the message of each C_i^{RS} can made equal. This is very useful for juxtaposing the codes together, as it makes sure that the dimension of each of the concatenated codes $\mathbf{C_i}$ will be the same.

The Inner Codes. The blocklength of each inner code C_i^{in} will be the same, say n_1. Note that this ensures that each one of the concatenated codes $\mathbf{C_i}$ has identical blocklength $N \stackrel{\mathrm{def}}{=} n_0 n_1$. The dimension of C_i^{in} will be m_i, so that it can be concatenated with the Reed-Solomon code C_i^{RS} (that was defined over $\mathrm{GF}(q_0^{m_i})$). The code C_i^{in} will have the properties guaranteed by Corollary 9.9 – specifically, it will have rate $r_i = m_i/n_1 = \Omega(\delta_{i-1})$ and will be

$((1-\delta_{i-1})n_1, O(1/\delta_{i-1}))$-list decodable.[7] This implies that the blocklength n_1 equals

$$n_1 = O\Big(\frac{m_i}{\delta_{i-1}}\Big) = O\Big(\frac{m\delta_i}{\delta_0\delta_{i-1}}\Big) = O\Big(\frac{m\Delta}{\varepsilon}\Big)\ ,$$

which is independent of i and can thus be made identical for each of the inner codes C_i^{in}.

The construction time of $\mathbf{C_i}$ is dominated by the construction time for the inner code C_i^{in}. Now, using Lemma 9.13, we know that a C_i^{in} with the required properties can be constructed in deterministic $q_0^{O(n_1)} = q_0^{O(\frac{m\Delta}{\varepsilon})}$ time. Alternatively, a construction that works with high probability can be obtained in $O(n_1^2 \log q_0) = O(m^2\Delta^2\varepsilon^{-2}\log(1/\varepsilon))$ time.

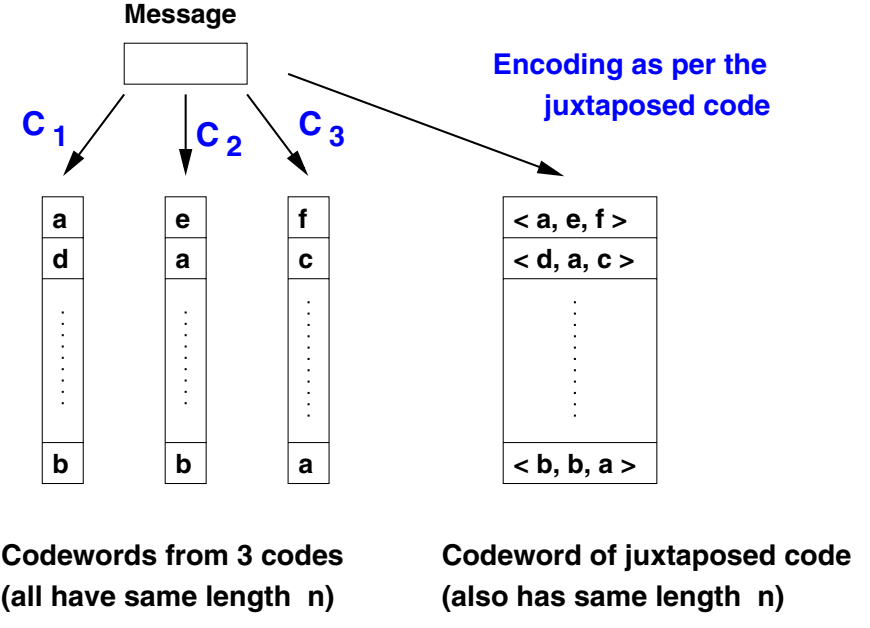

Fig. 9.6. Illustration of juxtaposition of three codes C_1, C_2, C_3.

The Juxtaposed Code. The final code, call it C^*, will be the juxtaposition of the codes $\mathbf{C_1}, \mathbf{C_2}, \ldots, \mathbf{C_t}$. Formally, by this we mean that to encode a message according to C^*, we will encode it according to each $\mathbf{C_i}$ to give a

[7] The result of Corollary 9.9 will give such codes over an alphabet of size $O(1/\delta_{i-1}^a)$ for any $a > 1$, but it can be checked that it will equally work over the larger alphabet $\mathrm{GF}(q_0)$ — essentially the larger alphabet only helps that result.

codeword, say $\langle c_{i1}, \ldots, c_{iN} \rangle \in \mathrm{GF}(q_0)^N$. The associated codeword of C^* will then be $\langle d_1, \ldots, d_N \rangle \in \mathrm{GF}(q_0^t)^N$ where $d_j = \langle c_{1j}, c_{2j}, \ldots, c_{tj} \rangle$ is interpreted as an element of $\mathrm{GF}(q_0^t)$. Figure 9.6 illustrates the juxtaposition operator applied to three codes.

We now pick parameters δ_i's appropriately in the above scheme and obtain the following theorem. (We use the notation developed above freely in the proof of the theorem.)

Theorem 9.25. *For every $\varepsilon > 0$, every integer $t \geq 1$ and each $b > t$, there exists a code family with the following properties:*

(i) It has rate $\Omega(t^{-3}\varepsilon^{2+2/t})$ and is defined over an alphabet of size $O(1/\varepsilon^b)$.

(ii) A code of blocklength N in the family can be constructed in $N^{O(1/\varepsilon^{1+1/t})}$ time deterministically, and a construction that has the list decodability property (iii) below with high probability can be obtained in probabilistic $O(\varepsilon^{-(2+2/t)} \log^2 N)$ time.

(iii) A code of blocklength N belonging to the family can be list decoded from up to a fraction $(1 - \varepsilon)$ of errors in $N^{O(1/\varepsilon)}$ time using lists of size $O(t^2/\varepsilon^{1+1/t})$.

Proof: Let $\varepsilon > 0$ be given. Pick $q_0 = O(1/\varepsilon^c)$ to be a power of 2 where $c = b/t > 1$. Let us pick the δ_i's in geometric progression with $\delta_0 = \varepsilon/2$, $\delta_t = 1$, and $\delta_i/\delta_{i-1} = \Delta$ for $1 \leq i \leq t$. Note that this implies $\Delta = (2/\varepsilon)^{1/t}$.

For every large enough integer m, we now apply the construction C^* discussed above with parameters $(q_0, m, \varepsilon, t, \Delta)$.

The code C^* is then clearly defined over an alphabet of size $q_0^t = O((1/\varepsilon)^{ct}) = O(1/\varepsilon^b)$ Recall that C^* is the juxtaposition of t codes $\mathbf{C_i}$, $1 \leq i \leq t$, each of which is obtained by the concatenation of a rate R_i Reed-Solomon code C_i^{RS} with a rate r_i inner code C_i^{in}, where $R_i = \Theta(\frac{\varepsilon^2}{t^2 \delta_i \Delta})$ and $r_i = \Theta(\delta_{i-1})$. Therefore the rate of each $\mathbf{C_i}$ equals

$$R_i r_i = \Omega\Big(\frac{\varepsilon^2}{t^2 \delta_i \Delta} \cdot \delta_{i-1}\Big) = \Omega\Big(\frac{\varepsilon^2}{t^2 \Delta^2}\Big) \ . \tag{9.7}$$

Let K, N be the common dimension and blocklength respectively of the $\mathbf{C_i}$'s. The rate of the juxtaposed code C^* is $1/t$ times the rate of each $\mathbf{C_i}$ because of the juxtaposition operator and hence

$$R(C^*) = \Omega\Big(\frac{\varepsilon^2}{t^3 \Delta^2}\Big) = \Omega\big(t^{-3}\varepsilon^{2+2/t}\big) \ ,$$

as claimed in Part (i) of the theorem.

The dominant component in the construction of C^* is once again the construction of the inner codes C_i^{in} used in the concatenated codes $\mathbf{C_i}$. By the argument from the discussion preceding this theorem, we have that each C_i^{in} can be constructed in $q_0^{O(m\Delta/\varepsilon)}$ time deterministically, and

$O(m^2\Delta^2\varepsilon^{-2}\log(1/\varepsilon))$ time probabilistically. Since the overall blocklength $N = n_0 n_1 = q_0^m n_1$, we have $m \leq \log N/\log q_0$. Therefore the constructions times are $N^{O(\Delta/\varepsilon)} = N^{O(1/\varepsilon^{1+1/t})}$ for a deterministic construction, and $O(\Delta^2\varepsilon^{-2}\log^2 N) = O(\varepsilon^{-(2+2/t)}\log^2 N)$ for a probabilistic construction (that works with high probability). This proves Property (ii) claimed in the theorem.

It remains to prove the list decodability property of C^*. Specifically, we wish to prove that given a received word $\mathbf{r} \in \mathrm{GF}(q_0^t)^N$, we can output a list of all codewords of C^* that differ from $\mathbf{r}$ in at most $(1-\varepsilon)N$ positions, in $N^{O(1/\varepsilon)}$ time. Indeed let $\mathbf{c} = C^*(\mathbf{x})$ be a codeword of C^* that differs from $\mathbf{r}$ in at most a fraction $(1-\varepsilon)$ of places. Note that both $\mathbf{r}$ and $\mathbf{c}$ can be broken up into n_0 blocks of n_1 symbols each, corresponding to the n_0 inner encodings at the n_0 positions of the outer Reed-Solomon codes. (Here we are using the fact that all the Reed-Solomon codes C_i^{RS} and the inner codes C_i^{in} have the same blocklength, namely n_0 and n_1, respectively.)

Now comes the crucial part. Since the overall agreement between $\mathbf{c}$ and $\mathbf{r}$ is at least a fraction ε of symbols, a standard averaging argument implies that there exists a set B consisting of at least $\varepsilon n_0/2$ inner blocks, such that $\mathbf{c}$ and $\mathbf{r}$ agree on more than a fraction $\varepsilon/2 = \delta_0$ of symbols within each block in B. Now imagine partitioning the blocks in B into t parts P_i, $1 \leq i \leq t$, in the following way. The part P_i consists of all blocks in B for which the fractional agreement between the portions of $\mathbf{c}$ and $\mathbf{r}$ corresponding to that block lies in the interval $(\delta_{i-1}, \delta_i]$. One of these parts must have at least $|B|/t$ blocks. Let this part be P_{i^*}. Hence we conclude that there exists *some* i^*, $1 \leq i^* \leq t$, such that for at least a fraction $\frac{\varepsilon}{2t\delta_{i^*}}$ of the n_0 blocks, $\mathbf{c}$ and $\mathbf{r}$ agree on at least a fraction δ_{i^*-1} of positions within that block.

Now consider decoding the n_0 inner codes $C_{i^*}^{\mathrm{in}}$ corresponding to this i^* up to a radius of $(1-\delta_{i^*-1})$ errors (here we focus attention on and use only the i^*'th symbol from each of the N "juxtaposed" symbols from the received word $\mathbf{r}$). This can be accomplished by brute-force in a total of $n_0 q_0^{m\delta_{i^*}/\delta_0} = q_0^{O(m/\varepsilon)} = n_0^{O(1/\varepsilon)}$ time. By the property of $C_{i^*}^{\mathrm{in}}$, this decoding only outputs a list $L_j^{(i^*)}$ of $O(1/\delta_{i^*-1})$ codewords (or in other words Reed-Solomon symbols for the code $C_{i^*}^{\mathrm{RS}}$) for each of the blocks j, $1 \leq j \leq n_0$. By our choice of i^*, at least $\frac{\varepsilon n_0}{2t\delta_{i^*}}$ of these lists have the "correct" symbol of $C_{i^*}^{\mathrm{RS}}(\mathbf{x})$.

To finish the decoding, it suffices to be able to list decode $C_{i^*}^{\mathrm{RS}}$ with these lists $L_j^{(i^*)}$, $1 \leq j \leq n_0$, as input, and find *all* messages $\mathbf{x}$ such that $L_j^{(i^*)}$ contains the j'th symbol of $C_{i^*}^{\mathrm{RS}}(\mathbf{x})$ for at least $\frac{\varepsilon n_0}{2t\delta_{i^*}}$ values of j. We can now apply the list recovering algorithm for Reed-Solomon codes from Chapter 6 (specifically Theorem 6.21) to accomplish this decoding task in near-quadratic time. Specifically, this follows by applying Theorem 6.21 with the choice $n = n_0$, $k = k_{i^*} - 1 = O(n_0 \frac{\varepsilon^2}{t^2\Delta\delta_{i^*}}) = O(n_0 \frac{\varepsilon^2\delta_{i^*-1}}{t^2\delta_{i^*}^2})$, $\ell = O(1/\delta_{i^*-1})$ and $\alpha = \frac{\varepsilon}{2t\delta_{i^*}}$. It can be verified that the condition $\alpha > \sqrt{2k\ell/n}$ can be

satisfied with these setting of parameters. Moreover, by Theorem 6.21, the number of codewords output by the decoding algorithm will be $O(\sqrt{n\ell/k})$, which is $O(t\Delta/\varepsilon)$ for our choice of parameters.

Of course, the algorithm cannot know the value of i^* in the above description, but running the above decoding procedure for each $\mathbf{C_i}$, $1 \leq i \leq t$, will output a list of size at most $O(t^2\Delta/\varepsilon) = O(t^2\varepsilon^{-(1+1/t)})$ that includes all codewords that differ from the received word $\mathbf{r}$ in at most a fraction $(1-\varepsilon)$ of the positions. The decoding time is dominated by the time to decode the inner codes, which, as discussed earlier, takes $n_0^{O(1/\varepsilon)} = N^{O(1/\varepsilon)}$ time. This completes the proof of Property (iii) of the theorem as well. □

Comparison with Algebraic-geometric codes: Note that for $t \leq 3$, the result of Theorem 9.25 is incomparable to AG-codes, since it gets a better alphabet size than AG-codes (which work over alphabet size of $O(1/\varepsilon^4)$), but the rate is worse than ε^2. Thus the above codes give some new, interesting trade-offs for codes that can be list decoded in polynomial time from a fraction $(1-\varepsilon)$ of errors.

By picking a fine "bucketing" with $\Delta = 2$ and $t = \lceil \lg(2/\varepsilon) \rceil$ in the above theorem, we can achieve a rate very close to ε^2 though the alphabet size becomes quasi-polynomial in $1/\varepsilon$. This gives us the following result.

Corollary 9.26. *For every $\varepsilon > 0$, there is a code family with the following properties:*

(i) (Rate and alphabet size) It has rate $\Omega(\varepsilon^2 \log^{-3}(1/\varepsilon))$ and is defined over an alphabet of size $2^{O(\log^2(1/\varepsilon))}$.

(ii) (Construction complexity) A code of blocklength N in the family can be constructed in $N^{O(1/\varepsilon)}$ time deterministically, and a construction that has the list decodability property (iii) below with high probability can be obtained in probabilistic $O(\log^2 N/\varepsilon^2)$ time.

(iii) (List decodability) A code of blocklength N in the family can be list decoded from up to a fraction $(1-\varepsilon)$ of errors in $N^{O(1/\varepsilon)}$ time using lists of size $O(\varepsilon^{-1}\log^2(1/\varepsilon))$.

9.7 Notes

The concept of good list recoverable codes, which was crucial to most of our results in this chapter, also appears in the work on extractor codes by Ta-Shma and Zuckerman [184]. The terminology "list recoverable codes" itself was introduced for the first time by the author and Indyk in [81]. Ta-Shma and Zuckerman also analyze the list recoverability of random codes. However, their results are for general random codes and their proof makes use of the complete independence of all the codewords. The result of Lemma 9.6, which appears in [81], works for random pseudolinear codes and also gives bounds

for list recovering with constant-sized lists. The result from [184], as stated there, works for list size that depends on the blocklength of the code, since their target is a more general decoding situation when the input lists at each position could be of widely varying and potentially very large sizes. We, on the other hand, place a uniform upper bound on the size of each input list, and moreover are mainly interested in situations where this upper bound is a small fraction of the alphabet size of the code.

In recent years, there have been several papers which construct codes using expander-like graphs. Broadly, these use such graphs in two ways: either to construct the parity check matrix [171, 176, 201] or to redistribute symbols around in the encoding process [6]. Our codes constructions follow the spirit of the second approach, in the sense that we also use expander-like graphs (specifically dispersers) to distribute the symbols of the message. However, our constructions are more involved than the construction of [6], since we want to make the codes efficiently decodable. In particular there is a lot more algorithmic focus in our work than in [6].

There has also been work on sub-exponential time *unique* decoding algorithms. In particular, the algorithm of [203] can unique decode certain large distance binary codes in $2^{O(\sqrt{n})}$ time. In fact it was this algorithm that motivated our discussion in Section 9.4.4. The quest for an improved decoding time led us to the constructions using multi-concatenated codes that were discussed in Section 9.5. The use of a sequence of inner codes in order to decrease the decoding time by paying only a constant factor in the rate at each level appears to be novel to the constructions in Section 9.5. Subsequent work (using certain *extractors*) by Guruswami [79] achieves results similar to those of Section 9.5.2 with an explicit construction: specifically, explicit codes of rate $\varepsilon/\log^{O(1)}(1/\varepsilon)$ are constructed in [79] that have sub-exponential time list decoding algorithms for a fraction $(1-\varepsilon)$ of errors.

Except for the results of Section 9.6, the rest of the material discussed in this chapter appears in [81]. The results of Section 9.6 appear in [82].