Quasilinear time list-decodable codes for space bounded channels

Swastik Kopparty* Ronen Shaltiel† Jad Silbak‡

November 13, 2019

Abstract

We consider codes for space bounded channels. This is a model for communication under noise that was studied by Guruswami and Smith (J. ACM 2016) and lies between the Shannon (random) and Hamming (adversarial) models. In this model, a channel is a space bounded procedure that reads the codeword in one pass, and modifies at most a p fraction of the bits of the codeword.

Guruswami and Smith, and later work by Shaltiel and Silbak (RANDOM 2016), gave constructions of list-decodable codes with rate approaching $1 - H(p)$ against channels with space $s = c \log n$, with encoding/decoding time $\mathrm{poly}(2^s) = \mathrm{poly}(n^c)$.

In this paper we show that for every constant $0 \leq p < \frac{1}{2}$, and every sufficiently small constant $\epsilon > 0$, there are codes with rate $R \geq 1 - H(p) - \epsilon$, list size $\mathrm{poly}(1/\epsilon)$, and furthermore:

- Our codes can handle channels with space $s = n^{\Omega(1)}$, which is much larger than $O(\log n)$ achieved by previous work.
- We give encoding and decoding algorithms that run in time $n \cdot \mathrm{polylog}(n)$. Previous work achieved large and unspecified $\mathrm{poly}(n)$ time (even for space $s = 1 \cdot \log n$ channels).
- We can handle space bounded channels that read the codeword in any order, whereas previous work considered channels that read the codeword in the standard order.

Our construction builds on the machinery of Guruswami and Smith (with some key modifications) replacing some nonconstructive codes and pseudorandom objects (that are found in exponential time by brute force) with efficient explicit constructions. For this purpose we exploit recent results of Haramaty, Lee and Viola (SICOMP 2018) on pseudorandom properties of "t-wise independence + low weight noise" which we quantitatively improve using techniques by Forbes and Kelly (FOCS 2018).

To make use of such distributions, we give new explicit constructions of binary linear codes that have dual distance of $n^{\Omega(1)}$, and are also polynomial time list-decodable from relative distance $\frac{1}{2} - \epsilon$, with list size $\mathrm{poly}(1/\epsilon)$. To the best of our knowledge, no such construction was previously known.

Somewhat surprisingly, we show that Reed-Solomon codes with dimension $k < \sqrt{n}$, have this property if interpreted as binary codes (in some specific interpretation) which we term: "Raw Reed-Solomon Codes". A key idea is viewing Reed-Solomon codes as "bundles" of certain dual-BCH codewords.

*Department of Mathematics and Department of Computer Science, Rutgers University. Email: swastik.kopparty@gmail.com. Research supported in part by NSF grants CCF-1253886, CCF-1540634, CCF-1814409 and CCF-1412958, and BSF grant 2014359. Some of this research was done while visiting the Institute for Advanced Study.

†Department of Computer Science. University of Haifa, Email: ronen@cs.haifa.ac.il. Research supported by ISF grant 1628/17.

‡School of Computer Science, Tel Aviv University. Email: jadsilbak@gmail.com. Research supported by ERC starting grant 638121. Part of this work was done while at the University of Haifa, and supported by ISF grant 1628/17

1 Introduction

A longstanding open problem in coding theory is to construct binary list-decodable codes that achieve list-decoding capacity, with efficient encoding and list-decoding algorithms. We start with a definition of list-decodable codes. Thinking ahead, the definition of codes below is stated in terms of algorithmic properties of encoding and decoding (rather than combinatorial properties using distance and Hamming balls).

Definition 1.1 (Codes)**.** *For $z \in \{0,1\}^n$, let weight(z) denote the Hamming weight of z. Namely, weight$(z) = |\{i \in [n] : z_i \neq 0\}|$. We say that $\mathrm{Enc} : \{0,1\}^k \to \{0,1\}^n$ is an encoding function for a code that is:*

- ***decodable from** t **errors**, if there exists a function $\mathrm{Dec} : \{0,1\}^n \to \{0,1\}^k$ such that for every $m \in \{0,1\}^k$ and every $e \in \{0,1\}^n$ with weight$(e) \leq t$, $\mathrm{Dec}(\mathrm{Enc}(m) \oplus e) = m$.*
- *L-**list-decodable from** t **errors**, if the function Dec is allowed to output a list of size at most L, and for every $m \in \{0,1\}^k$ and every $e \in \{0,1\}^n$ with weight$(e) \leq t$, $\mathrm{Dec}(\mathrm{Enc}(m) \oplus e) \ni m$.*

The rate of a code is $R = \frac{k}{n}$.

We will be interested in codes for $t = pn$ errors, where $0 \leq p < \frac{1}{2}$ is a constant, and n is sufficiently large. The "list-decoding capacity" in this setup is $R = 1 - H(p)$, meaning that for every constant $\epsilon > 0$, and sufficiently large n, there exist L-list decodable codes for pn errors, with rate $R \geq 1 - H(p) - \epsilon$, and list size $L = \mathrm{poly}(1/\epsilon)$. Despite substantial effort, it is not known how to construct such codes with poly-time encoding algorithms (even if one does not insist on poly-time list-decoding).

It is known that codes with rate $R < 1 - H(p)$ must have exponential size lists. The best known uniquely decodable codes have rate $R \leq 1 - H(2p)$, and (unlike the case of list-decoding) the precise capacity of unique decoding is not completely understood.

Hamming versus Shannon scenarios. The list-decoding task of Definition 1.1 is in the "Hamming scenario" in which the codeword $z = \mathrm{Enc}(m)$ is corrupted by an "unbounded channel" $C(\cdot)$ which given z produces an arbitrary "error pattern" $e = C(z) \in \{0,1\}^n$ with *weight*$(e) \leq pn$, and the decoding algorithm is required to decode (or list-decode) given the "corrupted received word" $z \oplus C(z) = z \oplus e$.

The "Shannon scenario" considers a "restricted channel" C which prepares the "error pattern" $e \in \{0,1\}^n$ *without looking at the codeword* $z = \mathrm{Enc}(m)$. The most well known example is a *binary symmetric channel* (BSC), in which the error pattern $e \in \{0,1\}^n$ is sampled from a distribution which we denote by BSC_p^n, in which the bits $e_1, \ldots, e_n$ are independent, and each e_i is one with probability p. The capacity of such a channel is $R = 1 - H(p)$, and a long line of works give explicit codes matching capacity with efficient (and in fact linear time) encoding and (unique) decoding algorithms [GI05].

Many other "channel distributions" are considered, and in some of them (like "bursts of errors") the individual bits of e are not chosen independently, but rather by a process with "small space".

Computationally bounded channels. Note that in Shannon's scenario, channels produce an error pattern that *does not* depend on the codeword $z = \mathrm{Enc}(m)$, whereas in Hamming's scenario there is no restriction, and channels may choose the error pattern as an arbitrary function of the codeword $z = \mathrm{Enc}(m)$. A natural intermediate scenario (considered by Lipton [Lip94]) is to allow the channel $C(z) = e$ to choose the error pattern as a function of z (while insisting that *weight*$(e) \leq pn$), but restrict our attention to channels C from some complexity class.

In this paper (following [GS16, SS16]) we will consider *space bounded channels* which read $z = Enc(m)$ in one pass, using limited space. In this scenario it is helpful to consider *stochastic codes* in which encoding and decoding procedures are randomized.

We now explain why randomization helps. If we insist on the standard notion of *deterministic* encoding algorithms for codes, then the notion of codes for bounded channels coincides with the standard (combinatorial) notion of codes (and so we don't gain by restricting to bounded channels). More specifically, Let $\mathrm{Enc} : \{0,1\}^k \to \{0,1\}^n$ be some function, and let m, e be the "worst message and error pattern" for Enc in terms of "combinatorial list decoding". That is, that for some number L, a Hamming ball of radius pn around $\mathrm{Enc}(m) \oplus e$ contains more than L codewords. This in particular means that the code is not L-list decodable from pn errors. The constant function $C(z) = e$ is a channel from which L-list decoding is impossible. This channel has low complexity in any nonuniform complexity class (as e depends on n, but not on z). In other words, if encoding is deterministic, then the worse combinatorial attack has low complexity. This argument does not apply if encoding algorithms are randomized, and decoding is only guaranteed to succeed with high probability (as is the case in stochastic codes that are defined below).

1.1 Stochastic codes for space bounded channels

Guruswami and Smith [GS16] considered a notion of *stochastic codes* in which encoding is randomized.[1] In this framework, the encoding algorithm Enc also receives d random bits, and the encoding of a message m, is a random variable $X = \mathrm{Enc}(m, U_d)$ (where U_d denote the uniform distribution on d bits). The channel C receives the "codeword" X as input, and produces an error pattern $e = C(X)$. The decoding algorithm Dec receives the corrupted received word $X \oplus e = X \oplus C(X)$, and needs to decode (or list decode) with high probability (over the choice of the random coins of encoding and decoding algorithms). *We stress that the decoding algorithm* ***does not*** *need to receive the random coins of the encoder*. A formal definition follows:

Definition 1.2 (Stochastic codes for bounded channels [GS16]). *Let $\mathcal{C}$ be a class of functions from n bits to n bits. We say that* $\mathrm{Enc} : \{0,1\}^k \times \{0,1\}^d \to \{0,1\}^n$ *is an encoding function for a stochastic code that is:*

- ***decodable*** *for "channel class" $\mathcal{C}$, with success probability $1-\nu$, if there exists a (possibly randomized) procedure* $\mathrm{Dec} : \{0,1\}^n \to \{0,1\}^k$ *such that for every $m \in \{0,1\}^k$ and every $C \in \mathcal{C}$, setting $X = \mathrm{Enc}(m, U_d)$, we have that* $\Pr[\mathrm{Dec}(X \oplus C(X)) = m] \geq 1 - \nu$*, where the probability is over coin tosses of the encoding and decoding procedures.*
- *L-**list-decodable** for "channel class" $\mathcal{C}$, with success probability $1-\nu$, if the procedure* Dec *is allowed to output a list of size at most L, and* $\Pr[\mathrm{Dec}(X \oplus C(X)) \ni m] \geq 1 - \nu$*, where the probability is over coin tosses of the encoding and decoding procedures.*

The rate of a stochastic code is $R = \frac{k}{n}$.

Following [GS16, SS16] we will be interested in the class $\mathcal{C}$ of functions that are computable in one pass using small space. This is captured by the model of oblivious read once branching programs (ROBP). Loosely speaking, a space s ROBP, $C : \{0,1\}^n \to \{0,1\}^n$ is a model of computation that on input $x \in \{0,1\}^n$ performs the following: The ROBP has an internal state q of s bits (initiated to zeros). At step i, the ROBP reads x_i and uses a "transition function" $\delta_i : \{0,1\}^s \times \{0,1\} \to \{0,1\}^s \times \{0,1\}$ to update its "internal state", and output a bit. Overall, on input $x \in \{0,1\}^n$, C produces an output $C(x)$ of n bits. (We sometimes need to consider ROBPs that output a single bit, and in this case the ROBP "decides" on its output bits as a function of its final state). A (more general) precise definition of ROBPs is given in Section 3.2.

[1] In the definition below we also allow the decoding algorithm to be randomized. This allows us to speed up decoding algorithms in some cases. Previous work of [GS16, SS16] did not use randomized decoding algorithms, partly because the running time of decoding was inherently large, and there was no gain in using randomness to speed it up.

We will consider channels that induce pn errors and are computable by space s ROBPs that may choose to read the bits of the codeword z in any order. This is captured by allowing the ROBP to permute the order of the bits using some permutation $\sigma : [n] \to [n]$ prior to reading it. We also "un-permute" the output bits, so that the output is presented in the standard order. The definition below is restated more formally in Definition 3.2.

Definition 1.3 (Space bounded channels, informal). *We say that a function $C : \{0,1\}^n \to \{0,1\}^n$ induces t errors if for every $z \in \{0,1\}^n$, weight$(C(z)) \le t$.*

*The class of **space** s **channels** is the class of functions $C : \{0,1\}^n \to \{0,1\}^n$ computed by space s ROBPs.*

*Let $\sigma : [n] \to [n]$ be a permutation, and for $z \in \{0,1\}^n$, let $\sigma(z)$ denote the n bit string z', in which $z'_i = z_{\sigma(i)}$. The class of **any-order space** s **channels** is the class of functions from n bits to n bits, of the form $e_C^\sigma = \sigma^{-1} \circ C \circ \sigma$ where $\sigma : [n] \to [n]$ is a permutation, and $C : \{0,1\}^n \to \{0,1\}^n$ is a space s ROBP.*

1.2 Our Results

1.2.1 New constructions of stochastic codes for space bounded channels

Guruswami and Smith [GS16] (and later work by Shaltiel and Silbak [SS16]) gave constructions of stochastic codes for space $s = O(\log n)$ channels with rate approaching $1 - H(p)$. However, a significant drawback of these works is that when set up against channels with space $s = c \log n$ for some constant c, the running time of encoding and decoding in [GS16, SS16] is polynomial in n, for a polynomial that is significantly larger than $2^s = n^c$. This means that one has to pay severely in efficiency, even when considering channels with moderate space.[2]

Guruswami and Smith [GS16] posed the open problem of removing this dependence, and coming up with a code for space $s = c \log n$ channels that has encoding and decoding that run in time n^{c_0} where c_0 is a universal constant, and does not grow with c. In this paper we *solve* this open problem, and in fact, go much farther. Our techniques give explicit constructions of stochastic codes with rate approaching $1 - H(p)$, and the following additional improvements:

- Our codes can handle channels with space $s = n^{\Omega(1)}$, which is much larger than $O(\log n)$ achieved by previous work.
- We give encoding and decoding algorithms that run in time $n \cdot \text{polylog}(n)$. Previous work achieved large and unspecified $\text{poly}(n)$ time (even for space $s = 1 \cdot \log n$ channels).
- Our success probability is $1 - \nu$ for $\nu = 2^{-\log^{O(1)} n} = n^{-\omega(1)}$, whereas previous works could only achieve $\nu = n^{-O(1)}$.
- We can handle any-order channels, whereas previous work considered channels that read the codeword in the standard order.

[2] We remark that the construction of Guruswami and Smith [GS16] is a "Monte-Carlo construction", meaning that it requires a preprocessing stage, in which a random string of length $\text{poly}(n^c)$ is shared between the encoding and decoding algorithm. The correctness of encoding and decoding algorithms is guaranteed w.h.p. over the choice of this string. (This string need not be kept secret from the channel, but note that this string is longer than the "description length" of the channel). In the final version of [GS16] it is observed that this Monte-Carlo approach can be extended to any class of channels where all channels have description length $\text{poly}(n^c)$, like for example size n^c circuits. Shaltiel and Silbak removed the need for a "Monte-Carlo" construction, and gave a construction that does not require this preprocessing step. However, their construction still suffers from running time of encoding and decoding that is exponential in $s = c \log n$. More details are given in Section 1.3.

These improvements are summarized in the following theorem (which is our main result). A more general version with more precise description of the dependencies between parameters is stated in Theorem 6.1.

Theorem 1.4 (quasilinear time codes for space $n^{\Omega(1)}$ channels)**.** *For every constant $0 \leq p < \frac{1}{2}$ and sufficiently small constant $\epsilon > 0$, there is an infinite family of stochastic codes with rate $R \geq 1 - H(p) - \epsilon$, that are $(L = \mathrm{poly}(1/\epsilon))$-list decodable for any-order space $s = n^{\Omega(1)}$ channels that induce pn errors, with success probability $1 - 2^{-\log^2 n}$. Furthermore, encoding and decoding run in time $n \cdot polylog(n)$.*

Our approach builds on the approach of [GS16] (and refinements of [SS16]) with some modifications and simplifications. The key to our improvements is a better explicit construction of some component that we call "control code" (for which [GS16, SS16] gave constructions based on showing existence by a non-constructive argument, and then finding the object by brute force search). We replace these inefficient arguments by an explicit construction. We give a detailed high level overview of the proof of Theorem 1.4 in Section 2.

We can also handle channels with space $s = n/\mathrm{polylog}(n)$, but we only know how to do this for small values of p, and then, encoding and decoding run in polynomial time (rather than quasilinear time). A more general version of the next theorem is stated in Theorem 6.3.

Theorem 1.5 (polynomial time codes for space $n/\mathrm{polylog}(n)$ channels)**.** *There exist constants $p_0 > 0$ and $c_0 \geq 1$ such that for every constant $0 \leq p < p_0$ and sufficiently small constant $\epsilon > 0$, there is an infinite family of stochastic codes with rate $R \geq 1 - H(p) - \epsilon$ that are $(L = \mathrm{poly}(1/\epsilon))$-list decodable for any-order space $s = \frac{n}{(\log n)^{c_0}}$ channels that induce pn errors, with success probability $1 - 2^{-\log^2 n}$. Furthermore, encoding and decoding run in time $\mathrm{poly}(n)$.*

Perspective. Our results clearly extend to any channel that is a convex combination of any-order space s channels. Furthermore, with an additional $\log n$ space, a channel can count the number of error that it induces, and avoid inducing more than pn errors. This means that our theorems handle any distribution over any-order space s channels in which the probability of inducing significantly more than pn errors is small.

It was pointed out by Guruswami and Smith [GS16] that all the "stochastic channels" studied in Shannon's scenario are captured by this framework. Consequently, Theorem 1.4 can be seen as providing a *unified* solution that handles all such channels with rate approaching $1 - H(p)$ and quasilinear time encoding and decoding.

On a more philosophical level, one may postulate that the behavior of most conceivable channels that are not "fully adversarial" is captured by this framework of Guruswami and Smith, which can now be implemented in quasilinear time (without the severe penalty of the dependence of running time on the space of the channel).

1.2.2 Raw Reed-Solomon Codes

One of the tools that we require in order to prove our main theorem, is a *binary linear code* $\mathrm{Enc} : \mathbb{F}_2^k \to \mathbb{F}_2^n$ with the following properties.[3]

- Distance $(1/2 - o(1)) \cdot n$.

[3] In this section we use a more standard notation of coding theory. With our notation, a binary linear code is a code that has a linear encoding function $\mathrm{Enc} : \mathbb{F}_2^k \to \mathbb{F}_2^n$. The image of this function is a subspace C of the vector space $\mathbb{F}_2^n$. The **dual code** is the dual subspace $C^{\perp} = \{v \in \mathbb{F}_2^n : \forall c \in C, < v, c >= 0\}$. The **dual distance** is the minimum distance of $C^{\perp}$. The precise standard definitions are given in Section 3.5.2.

- Large dual distance of at least $n^{\Omega(1)}$. (In fact, we need a slightly stronger property to be explained later).
- Polynomial time list-decoding with list size $\mathrm{poly}(1/\epsilon)$ from $(\frac{1}{2}-\epsilon)\cdot n$ errors, for every sufficiently small constant $\epsilon > 0$. (This in particular implies poly-time unique decoding up $(\frac{1}{4}-\epsilon)\cdot n$ errors, by pruning the list and keeping only the unique codeword that is closest to the received word).

To the best of our knowledge, no construction with these properties is known. In this paper we exhibit such codes. Interestingly, we show that for some settings of parameters, Reed-Solomon codes have these properties if interpreted as binary codes suitably. We call these resulting binary codes *Raw Reed-Solomon codes*.

More specifically, let m be an integer, and $n_{\mathrm{RS}} = 2^m - 1$. We consider the field $\mathbb{F}_{2^m}$, and Reed-Solomon codes of degree $\leq d$ with n_{RS} evaluation points given by $D = \mathbb{F}_{2^m} \setminus \{0\}$. That is, the encoding of a message $w \in \mathbb{F}_{2^m}^{d+1}$, is $\mathrm{Enc}(w) = (\sum_{0 \leq i \leq d} w_i \cdot t^i)_{(t \in S)}$. This code has distance $n_{\mathrm{RS}} - d$, and alphabet size $q_{\mathrm{RS}} = 2^m$. It can be interpreted as a binary linear code by choosing some $\mathbb{F}_2$-linear bijection $\Phi : \mathbb{F}_{2^m} \to \mathbb{F}_2^m$ (which is used to interpret field elements as m bit vectors), and applying Φ on each of the n_{RS} symbols of the codeword. This gives a binary linear code $\mathsf{VerySimpleRawRS}$ with dimension $k = (d+1)\cdot m$ and block length $n = n_{\mathrm{RS}} \cdot m$. It immediately follows that this standard construction has distance at least $n_{\mathrm{RS}} - d$, but note that in terms of *relative distance*, this quite general argument does quite poorly, since it can never show that the relative distance is more than:

$$\frac{n_{\mathrm{RS}}}{n} = \frac{1}{m} = \Theta\left(\frac{1}{\log n}\right) = o(1).$$

In fact, this is the truth, and $\mathsf{VerySimpleRawRS}$ truly does have $o(1)$ relative distance.

Nevertheless, we show that a slight modification of this code has extremely good distance (and keeps the dual distance). Let $\mathsf{SimpleRawRS}$ be the subcode of $\mathsf{VerySimpleRawRS}$ which only includes codewords that come from polynomials which have 0 constant term. Using deep algebraic tools (very specific to the algebraic situation at hand) we show that if the degree bound $d < n_{\mathrm{RS}}^{o(1)}$ (so that the dimension k satisfies $k = n^{o(1)}$), then $\mathsf{SimpleRawRS}$ has relative distance $\frac{1}{2} - o(1)$. We also define another variant, $\mathsf{OddRawRS}$, which has the same relative distance but which can achieve any dimension that is $o(n^{1/2})$. Finally, using the powerful algorithmic decoding algorithms known for Reed-Solomon codes, we show that these codes are also list-decodable.

A more detailed description of Raw Reed Solomon codes appears in Section 4. In particular we prove the following theorem.

Theorem 1.6 (Codes with large distance and dual distance)**.** *For every constant $0 < \alpha < 1/2$, and every sufficiently large m, setting $n = (2^m - 1)\cdot m$, and $k = n^\alpha$, there is a binary linear $[n,k]_2$-code C that satisfies:*

- *C has distance $(\frac{1}{2} - O((\frac{\log n}{n})^{\frac{1}{2}-\alpha}))n = (\frac{1}{2} - o(1))\cdot n$.*
- *C has dual distance $\Omega(\frac{n^\alpha}{\log n})$.*
- *C has a linear encoding map $\mathrm{Enc} : \mathbb{F}_2^k \to \mathbb{F}_2^n$ that runs in time $\mathrm{poly}(n)$.*
- *There exists a universal constant b, such that for every $\epsilon \geq b\sqrt{\alpha}$, Enc is $O(\frac{1}{\epsilon^2})$-list-decodable from $(\frac{1}{2}-\epsilon)n$ errors in time $\mathrm{poly}(n)$.*

A key property that we use in the analysis is that (modulo some caveats) a codeword of length $n_{\text{RS}} \cdot m$ of the $\mathsf{SimpleRawRS}$ and $\mathsf{OddRawRS}$ codes can be viewed as the juxtaposition of m binary strings of length n_{RS} in a natural way. These m binary strings turn out to be (correlated) codewords of a dual-BCH code, and our analysis of the distance exploits this. This is explained more precisely is Section 2.

Dual-BCH codes themselves satisfy the first three requirements above, but they are not known to have efficient decoding. Curiously, our result shows that a code of correlated tuples of dual-BCH codewords *can* be decoded efficiently, while retaining the other good properties of dual-BCH codes.

Reed Solomon codes have a lot of structure and many useful properties (in addition to their distance properties) and so, we believe that the fact that Raw Reed-Solomon codes have the additional properties listed above (when viewed as binary codes) is of independent interest, and may prove useful in other applications.

1.3 More related work on codes for bounded channels

1.3.1 Stochastic codes for other classes of channels

Additive channels. Guruswami and Smith [GS16] gave constructions of stochastic codes with rate approaching $1 - H(p)$ that are uniquely decodable for additive channels that induce pn errors, with success probability $1 - 2^{-\Omega(n/\log n)}$. In our notation these are the constant functions $C(\cdot) = e$ where e is a constant string with weight at most pn. The encoding and decoding algorithms in [GS16] run in polynomial time. Our approach can be used in this setup, and can speed up the encoding and decoding algorithms to run in quasilinear time, if the success probability is reduced to $1 - 2^{-\text{polylog}(n)}$.

poly-size circuits and bounded space channels. In the same paper, Guruswami and Smith also gave constructions of stochastic codes with rate approaching $1-H(p)$ that are list-decodable for space $s = c \log n$ channels (or size n^c circuits) that induce pn errors, with success probability n^{-c}. As explained earlier in Section 1.2.1, a significant drawback of these results is that the running time of the encoding algorithm was polynomial in n^c, for a large and unspecified polynomial (meaning that efficiency quickly deteriorates even for conservative estimates on channel complexity). The construction of [GS16] is "Monte-Carlo". Meaning that it requires a preprocessing stage, in which a random string of length $\text{poly}(n^c)$ is shared between the encoding and decoding algorithm. The correctness of encoding and decoding algorithms is guaranteed w.h.p. over the choice of this string. (This string need not be kept secret from the channel).

Shaltiel and Silbak [SS16] removed the need for a preprocessing stage by slightly modifying the construction of Guruswami and Smith, and providing explicit constructions for the modified components. They give results for space $s = c \log n$ channels, and size n^c circuits (here a complexity assumption that there are functions in $\mathit{DTIME}(2^{O(n)})$ that are hard for small circuits is used, and is necessary). Shaltiel and Silbak also consider channels that are implementable by constant depth circuits, and provide constructions of stochastic codes for this setup.

1.3.2 Other coding scenarios with randomized encoding/bounded channels

The notion of computationally bounded channels was also studied in other setups. We mention some of these works below.

Shared private randomness. We start with the notion of codes with "shared private randomness". While this setup was considered before the notion of stochastic codes, in this paper, it is natural to view it as a version of stochastic codes in which the decoding algorithm *does* receive the randomness S chosen by the encoding algorithm. This corresponds to a standard symmetric cryptography setup in which honest parties (the encoder and decoder) share a uniform private key S, and the bad party (the channel) does not get the key. Lipton [Lip94] and following work (see [Smi07] for more details) gave explicit constructions of uniquely decodable codes against computationally bounded channels, in this setup, with rate approaching $1 - H(p)$, under cryptographic assumptions.

Note that the setup of stochastic codes is lighter. The encoder and decoder do not need to share a private random key. Moreover, a fresh key can be chosen on the spot every time the encoder encodes a message.

A related notion of "private codes" was studied by Langberg [Lan04]. This is also in the setup of shared private randomness. Here channels are computationally unbounded, codes are existential rather than explicit, and have rate approaching $1 - H(p)$. The focus is on minimizing the length of the shared key. Langberg provides asymptotically matching upper and lower bounds of $\Theta(\log n + \log(1/\nu))$, on the amount of randomness that needs to be shared for unique decoding in this setup, where ν is the error parameter.

Non malleable codes. Non-malleable codes (introduced by Dziembowski, Pietrzak, and Wichs [DPW18]) consider channels that are not restricted in the number of errors that they induce. Instead, channels are assumed to come from some limited class of functions (or complexity class). Codes are stochastic (meaning that the encoding procedure is randomized) and it is required that following the corruption by the channel, the decoder either reproduces the encoded message, or an "unrelated" message. The definition of "unrelated" is given using the simulation paradigm from cryptography. Several classes have been considered, and some of the constructions rely on cryptographic assumptions. The reader is referred to [DPW18] and the references therein for precise definition and a survey of results in non-mallable codes.

Encoding a uniform message. Haviv and Langberg [HL11] consider a model where encoding and decoding is deterministic, but the message to be encoded is chosen uniformly at random. They show the existence of codes in this setup, that have unique decoding, and beat the Gilbert-Varshamov bound.

Public key setup. Micali et al. [MPSW10] considered computationally bounded channels, and a cryptographic public key setup. Their focus is to use this setup to convert a given (standard) explicit list-decodable code into an explicit uniquely decodable codes (in this specific public key setup).

1.4 Organization of the paper

In Section 2 we give a high level overview of the ideas and techniques in this paper. In Section 3 we give definitions and past work on the tools and ingredients that are used in our construction. In Section 4 we state and prove our results on raw Reed-Solomon codes. In Section 5 we use raw Reed-Solomon codes to construct stochastic control codes. In Section 6 we give our main construction of stochastic codes for space bounded channels (which relies on stochastic control codes). In Section 7 we prove the correctness of our main construction.

2 Overview of the technique

In this section we give a high level overview of the ideas and techniques that we use. We allow ourselves to be informal and imprecise (in order to highlight the main ideas). Complete definitions, theorem statements and proofs, appear in later sections (which do not rely on the informal description given in this section).

2.1 Stochastic control codes

The construction of codes for bounded channels of Guruswami and Smith [GS16], as well as later modification by Shaltiel and Silbak [SS16] use a component which we call a "stochastic control code".

Definition 2.1 (Stochastic control code, informal)**.** *A function* $\mathrm{Enc}_{\mathrm{ctrl}} : \{0,1\}^k \times \{0,1\}^d \to \{0,1\}^n$ *is a stochastic control code that is:*

- *pseudorandom, if for every* $x \in \{0,1\}^k$, $\mathrm{Enc}_{\mathrm{ctrl}}(x, U_d)$ *is pseudorandom for small space ROBPs.*
- *List decodable, if for every sufficiently small constant* $\epsilon > 0$*, there is an explicit list-decoding algorithm* $\mathrm{Dec}_{\mathrm{ctrl}}$ *with constant size lists, such that for every* $x \in \{0,1\}^k$*,* $y \in \{0,1\}^d$*, and* $e \in \{0,1\}^n$ *with* $\mathsf{weight}(e) \le (\frac{1}{2} - \epsilon) \cdot n$*, list-decoding succeeds, that is,* $x \in \mathrm{Dec}_{\mathrm{ctrl}}(\mathrm{Enc}_{\mathrm{ctrl}}(x, y) \oplus e)$*.*

A more precise definition (with precise quantities) appears in Section 5. The construction of capacity achieving stochastic codes for space bounded channels (Theorem 1.4) will rely on stochastic control codes that are pseudorandom and list-decodable. Definition 2.1 requires recovery from *adversarial* errors (that may be induced by an *unbounded* channel) while also requiring the additional pseudorandomness property. This makes the requirements *stronger* than the codes for bounded space channels that we aim to construct.

A key idea is that in the final construction, the control code will be used to encode a short "control string", and so, the rate of this code does not need to approach $1 - H(p)$, and we will be able to use control codes in which $k = n^{\Omega(1)}$ in our final stochastic codes for bounded channels, and still have rate approaching $1 - H(p)$.

Theorem 2.2 (Control code, informal)**.** *There is an explicit control code* $\mathrm{Enc}_{\mathrm{ctrl}} : \{0,1\}^{n^{\Omega(1)}} \times \{0,1\}^{O(n)} \to \{0,1\}^n$ *which satisfies Definition 2.1.*

We start by explaining this construction, and later show how to use it to obtain stochastic codes for small space channels with rate approaching $1 - H(p)$.

2.1.1 Raw Reed-Solomon codes

The first step in our construction of control codes, are explicit (standard) binary linear codes with large dual distance and poly-time list decoding. These are the codes stated in Theorem 1.6. For concreteness, let m be an integer, and consider the field $\mathbb{F}_{2^m}$. Let $D = \mathbb{F}_{2^m} \setminus \{0\}$. We will consider evaluations of polynomials of degree at most d with $\mathbb{F}_{2^m}$ coefficients at the points of D, and then convert these evaluations to binary vectors of length m using an $\mathbb{F}_2$-linear bijection $\Phi : \mathbb{F}_{2^m} \to \mathbb{F}_2^m$. More generally, this conversion to binary vectors can also be done using a different $\mathbb{F}_2$-linear bijection Φ_x at each point $x \in D$. Overall, this gives us binary codewords of length $n = m \cdot (2^m - 1)$. We call the codes obtained this way *Raw Reed-Solomon Codes*, RawRS.

In this high level overview we will explain the analysis of a particular Raw Reed-Solomon code which we call *Odd Raw Reed-Solomon codes*[4], $\mathsf{OddRawRS}$. This is an instance of the above RawRS family of codes (for a particular choice of Φ_x), but it has a more direct description which we give next. Let $k = m \cdot (d+1)$ and $n = m \cdot (2^m - 1)$. Given a message $w \in \mathbb{F}_2^k$, we break it into $d+1$ blocks of m bits each, use these m-bit blocks to specify elements $\gamma_0, \gamma_1, \ldots, \gamma_d \in \mathbb{F}_{2^m}$, and consider the polynomial $P(X) = \sum_{j=0}^{d} \gamma_j X^{2j+1}$ which has *only odd degree monomials*. The codeword $c : D \times [m] \to \mathbb{F}_2$

[4] A similar but more involved analysis applies to the more natural code $\mathsf{SimpleRawRS}$.

corresponding to this message w is then the n bit long string obtained by taking all the evaluations of P and writing them in bits using Φ:

$$c(x,i) = \Phi(P(x))_i.$$

A key observation is that this interpretation is closely related to the dual-BCH code [MS77]. Dual-BCH codes are known to satisfy the first three properties in Theorem 4.1. That is they have poly-time encoding, large distance, and large dual distance. However, they are not known to have poly-time decoding or list-decoding.

We show that $\mathsf{OddRawRS}$ codes satisfy all four requirements. First we elaborate on the connection to dual-BCH codes. For each $i \in [m]$, consider the function $c_i : D \to \mathbb{F}_2$ given by $c_i(x) = c(x,i)$ (this is just a subset of the bits of c). Then each c_i is a codeword of the dual-BCH code. More specifically, the dual BCH codeword $c^{\mathsf{dual-BCH}} : D \to \mathbb{F}_2$ that corresponds to $w = (\gamma_0, \ldots, \gamma_d)$, can be defined as $c^{\mathsf{dual-BCH}}(x) = \mathrm{Tr}(\sum_{j=0}^{d} \gamma_j \cdot x^{2j+1})$, where $\mathrm{Tr} : \mathbb{F}_{2^m} \to \mathbb{F}_2$ is the $\mathbb{F}_2$-linear map that is the *field trace*. Furthermore, any bijective linear map $\Phi : \mathbb{F}_{2^m} \to \mathbb{F}_2^m$ can be expressed as m $\mathbb{F}_2$-linear maps $\Phi_i : \mathbb{F}_{2^m} \to \mathbb{F}_2$, where each Φ_i is defined by $\Phi_i(x) = \mathrm{Tr}(b \cdot x)$, for some nonzero $b \in \mathbb{F}_{2^m}$. It follows that $c_i(x) = \mathrm{Tr}(\sum_{j=0}^{d}(b \cdot \gamma_j) \cdot x^{2j+1})$, which can be viewed as the dual-BCH encoding of the nonzero word $(b \cdot \gamma_0, \ldots, b \cdot \gamma_d)$.

Thus the codewords of $\mathsf{OddRawRS}$ are just a sequence of correlated nonzero dual-BCH codewords. Using this connection to dual-BCH codes we get that $\mathsf{OddRawRS}$ codes have large distance.[5] This part of the argument does not work for general RawRS codes.

The remaining three properties hold for general RawRS codes. Efficient encoding is clear. The dual distance of $\mathsf{OddRawRS}$ codes follows from the dual distance of a related Reed-Solomon code, and the fact that Φ is a bijection.

Finally we come to the decodability. This is where we go beyond what is known for dual-BCH codes. The crucial point here is the connection to Reed-Solomon codes, for which amazing decoding algorithms are known [Sud97, GS99]. We show that the natural 2-stage list-decoding algorithm for $\mathsf{OddRawRS}$ (which is naturally viewed as a concatenated code) indeed decodes from $(1/2-\epsilon)$-fraction errors. The first stage is list-decoding of the inner blocks, which leads to a huge list of candidate symbols for each coordinate (since the inner blocks are all codes with minimum distance only 1). Then the efficient list-recovery algorithms known for Reed-Solomon codes, which can handle huge lists and $(1-o(1))$-fraction error, enables us find a large list that contains all nearby codewords. This implies the unique decodability from $(1/4-\epsilon)$-fraction errors. Finally, for list-decodability, using the fact that $\mathsf{OddRawRS}$ has relative distance $1/2 - o(1)$, the Johnson bound [Joh62] implies that the list of $(1/2-\epsilon)$-fraction close codewords is in fact $\mathrm{poly}(1/\epsilon)$, and we get the desired list-decoding algorithm.

2.1.2 From Raw Reed-Solomon codes to stochastic control codes.

We now explain how to prove Theorem 2.2 using $\mathsf{OddRawRS}$ codes (and specifically, the code stated in Theorem 1.6). This approach is inspired by a related argument that was used by Shaltiel and Silbak [SS16] to construct control codes against AC^0 circuits.

We will construct the control code $\mathrm{Enc}_{\mathrm{ctrl}} : \{0,1\}^{k/2} \times \{0,1\}^{d=k/2+n\cdot\log(1/\eta)} \to \{0,1\}^n$ as follows: Given $x \in \{0,1\}^{k/2}$, $r \in \{0,1\}^{k/2}$ and $v \in \{0,1\}^{n\cdot\log(1/\eta)}$. We use v as random coins to sample an element from BSC_η^n (that is n i.i.d. coins that evaluate to one with probability η) and define:

$$\mathrm{Enc}_{\mathrm{ctrl}}(x;(r,v)) = \mathrm{Enc}_{\mathsf{OddRawRS}}(r \circ x) \oplus \mathrm{BSC}_\eta^n.$$

[5] In the $\mathsf{OddRawRS}$ case it is almost immediate, in the $\mathsf{SimpleRawRS}$ case we need to understand the correlations between the component dual-BCH codewords.

That is, we use the linear code from the previous section to encode $r \circ x$ and xor the output with BSC_η^n. By Theorem 1.6 the $\mathsf{OddRawRS}$ code has dual distance $t = n^{\Omega(1)}$, and this can be used to show that for every $x \in \{0,1\}^{k/2}$, $\mathrm{Enc}_{\mathsf{OddRawRS}}(U_{k/2} \circ x)$ is $(t-1)$-wise independent.

Actually, for this to hold we need a stronger property, namely that if we consider the code $\mathrm{Enc}^{\mathrm{trunc}} : \mathbb{F}_2^{k/2} \to \mathbb{F}_2^n$ defined by $\mathrm{Enc}^{\mathrm{trunc}}(r) = \mathrm{Enc}_{\mathsf{OddRawRS}}(r \circ 0^{k/2})$, then this linear code has dual distance t. (This stronger property also holds for $\mathsf{OddRawRS}$ codes). Once we have that, by linearity, $\mathrm{Enc}_{\mathsf{OddRawRS}}(U_{k/2} \circ x) = \mathrm{Enc}^{\mathrm{trunc}}(U_{k/2}) \oplus L(x)$ where L is some linear function. Thus, it is enough to show that $\mathrm{Enc}^{\mathrm{trunc}}(U_{k/2})$ is $(t-1)$-wise independent. Note that encoding by $\mathrm{Enc}^{\mathrm{trunc}}$ is done by multiplying the message by the generator matrix of $\mathrm{Enc}^{\mathrm{trunc}}$ (which is the parity check matrix of the dual code). As the dual distance of $\mathrm{Enc}^{\mathrm{trunc}}$ is at least t, every $t-1$ columns of the latter matrix are linearly independent, and so $\mathrm{Enc}^{\mathrm{trunc}}(U_{k/2})$ is $(t-1)$-wise independent. It follows that the output distribution $\mathrm{Enc}_{\mathrm{ctrl}}(x, U_d)$ is a "t-wise independent distribution plus low weight BSC noise".

2.1.3 t-wise independence + low weight BSC noise.

Haramaty, Lee and Viola [HLV18] studied the pseudorandomness of such distributions, and showed that distributions of this form are pseudorandom for any-order small space ROBPs if t is sufficiently larger than $n^{2/3}$, and η is not too small (any constant $\eta > 0$ will do). Note that the list decoding property of Theorem 2.2 immediately follows for our construction (as $\mathsf{OddRawRS}$ codes have list decoding up to $(\frac{1}{2} - \epsilon) \cdot n$ errors, and so, if η is sufficiently small, the additional relative error of η can be "swallowed" in ϵ.

We now turn our attention to the pseudorandomness property of Theorem 2.2. Unfortunately, the dual distance t of $\mathsf{OddRawRS}$ codes cannot be larger than $\sqrt{n}$. This means, that at best, $\mathrm{Enc}_{\mathsf{OddRawRS}}(U_{k/2} \circ x)$ is t-wise independent for $t < \sqrt{n}$, whereas the results of [HLV18] give nothing unless $t \gg n^{2/3}$.

Recent work by Lee and Viola [LV17], and Forbes and Kelly [FK18], showed that "t-wise independence + *large weight* noise" *is* pseudorandom for small space any-order ROBPs even for small t (e.g., $t = O(s + \log n)$). However, these results use noise that is a conjunction of a t-wise independent distribution with a uniform distribution, and such noise has relative weight roughly $\frac{1}{4}$. This will not do for our list-decoding argument.

Fortunately, we can use the technique of Forbes and Kelly [FK18] to give a better analysis than Haramaty, Lee and Viola [HLV18] and show that $\mathrm{Enc}(x, U_d)$ is pseudorandom for any-order space $s = \Omega(t)$ ROBPs even for $t = n^{\Omega(1)}$. The precise statement appears in Theorem 5.6. This shows that $\mathrm{Enc}_{\mathrm{ctrl}}$ satisfies the properties in Definition 2.1 and proves Theorem 2.2.

2.2 The construction of stochastic codes for space bounded channels

In this section we give a sketch of the construction of stochastic codes for any-order space bounded channels. Our construction heavily builds on the machinery developed by Guruswami and Smith [GS16] (which in turn relies on previous ideas by Lipton [Lip94] and Smith [Smi07]). We also use the refinements of Shaltiel and Silbak [SS16], as well as several new modifications and simplification.

Recall that given a constant $\epsilon > 0$, our goal is to design a stochastic code $\mathrm{Enc} : \{0,1\}^{RN} \times \{0,1\}^d \to \{0,1\}^N$ that has rate $R \geq 1 - H(p) - \epsilon$, and is list-decodable for small space channels that induce pN errors. Furthermore, we aim for quasilinear time encoding and list-decoding with constant sized lists.

2.2.1 The encoding algorithm

To encode a message $m \in \{0,1\}^{RN}$ the encoding algorithm will encode m by a code $\mathrm{Enc}_{\mathrm{BSC}}$ for binary symmetric channels BSC_p. There are explicit constructions of such codes with rate approaching $R = 1 - H(p)$ and linear time encoding and decoding [GI05]. Thinking ahead, we set the block length of $\mathrm{Enc}_{\mathrm{BSC}}$ to be $N_{\mathrm{data}} = (1-\epsilon) \cdot N$, which we can do by choosing a smaller constant $\epsilon_{\mathrm{BSC}} = \epsilon/10$ for the BSC code. The encoding algorithm will also select random "seeds" to activate several "pseudorandom components". The seeds of all these components will be of length N^{α} for some small constant $\alpha > 0$, and so their length is negligible compared to N.

The first seed s_π will be used to generate an "almost t-wise independent permutation" $\pi : [N_{\mathrm{data}}] \to [N_{\mathrm{data}}]$. In this high level overview we will pretend that π is a random permutation. The encoding algorithm computes $x = \mathrm{Enc}_{\mathrm{BSC}}(m)$, and $y = \pi^{-1}(x)$ (recall that this means that the bits of x are "reordered" according to π^{-1}). Loosely speaking, this is done so that if the channel is an *additive channel* (namely one that has a fixed error pattern $e \in \{0,1\}^{N_{\mathrm{data}}}$ of weight pN) and if the decoding algorithm has a copy of s_π, then the decoding algorithm can apply π on the received word $y \oplus e$ and obtain $\mathrm{Enc}(m) \oplus \pi(e)$. For a random permutation π, the distribution $\pi(e)$ is very similar to BSC_p, and so the decoding algorithm can decode by applying $\mathrm{Dec}_{\mathrm{BSC}}$.

The argument above (which was suggested by Lipton [Lip94]) crucially requires that the decoding algorithm receives the seed s_π. The approach of Guruswami and Smith is to encode the seed s_π (as well as other seeds that we introduce soon) by a "control code" and "merge" y and this "control encoding" together, in the hope that the channel is not able to "wipe out" the control information, and furthermore, that the decoding algorithm is able to identify and correctly decode the control information.

For this purpose, the encoding algorithm also chooses a random seed s_{PRG} for a pseudorandom generator G that fools any-order small space ROBPs (we use the PRG of Forbes and Kelly [FK18]). When preparing the data part, the encoding xors the string $y = \pi^{-1}(\mathrm{Enc}_{\mathrm{BSC}}(m))$ with $G(s_{\mathrm{PRG}})$ to obtain the "data codeword" $c_{\mathrm{data}} = \pi^{-1}(\mathrm{Enc}_{\mathrm{BSC}}(m)) \oplus G(s_{\mathrm{PRG}})$. Loosely speaking, this means that c_{data} looks random to the channel.

The encoding algorithm now prepares the control codeword. For this purpose, the encoding algorithm divides the N output bits into $n = N^{1-\lambda}$ blocks of length $b = N^{\lambda}$, where $\lambda > 0$ is some small constant. It chooses an additional random seed s_{samp} for an "averaging sampler". This seed is used to specify $\epsilon \cdot n$ distinct indices $i_1, \ldots, i_{\epsilon \cdot n} \in [n]$. These blocks are called "control blocks", and the remaining blocks are called "data blocks". In this high level overview we pretend that the indices of control blocks are uniformly distributed in $[n]$. (Loosely speaking, the definition of averaging samplers allows us to make this assumption).

The final codeword $c \in \{0,1\}^N$ is prepared as follows: Note that the total length of data blocks is N_{data}, and the encoding algorithm "places" the data codeword c_{data} in these blocks. The remaining ϵn blocks are used to encode the "control information" $s = (s_\pi, s_{\mathrm{PRG}}, s_{\mathrm{samp}})$. This is done as follows: for each control block i, the encoding algorithm sets $c_i = \mathrm{Enc}_{\mathrm{ctrl}}(s, U_d)$ with fresh randomness for each block (where $\mathrm{Enc}_{\mathrm{ctrl}}$ is the stochastic control code of the previous section).[6] Note that this indeed gives a codeword c of length N.

[6] We mention that here we simplify previous work by [GS16, SS16] that also used an "outer control code" that was chosen to be list-recoverable. This simplification allows us to speed up the encoding and decoding as explained later.

2.2.2 The list decoding algorithm

In order to decode, the decoding algorithm first applies the list-decoding algorithm $\mathrm{Dec}_{\mathrm{ctrl}}$ on all the n blocks of the received word. The decoding algorithm obtains $\ell = \frac{n}{\epsilon^c}$ outcomes (where $c = 2$ is the exponent of the list size of $\mathrm{Dec}_{\mathrm{ctrl}}$), and it "passes on" each outcome that appears at least say $\epsilon^2 n$ times. (Note that there are at most $\mathrm{poly}(1/\epsilon)$ such outcomes). For each such candidate s' for control information, the decoding produces a message. Namely, it identifies the partition of blocks into control blocks and data blocks according to s'_{samp}. It then xors the data part with $G(s'_{\mathrm{PRG}})$, and applies the permutation π (defined by seed s'_{π}). Finally, it performs decoding by $\mathrm{Dec}_{\mathrm{BSC}}$. This process indeed produces a list of size $\mathrm{poly}(1/\epsilon)$ of candidate messages.

The analysis will show that for any small space channel, w.h.p. the "correct control information" s is one of the candidates s' considered by the decoding algorithm, and that with $s' = s$, the correct message m is decoded w.h.p.

2.2.3 Analyzing the construction

The analysis of the construction is quite involved and is presented in detail in Sections 6 and 7. On a high level, the key observation is that from the point of view of a space bounded channel, the data part looks random (as it is xored with the output of a pseudorandom generator) and each control block looks random (by the pseudorandom property of $\mathrm{Enc}_{\mathrm{ctrl}}$). This intuitively means that the channel cannot distinguish data blocks from control blocks, and therefore, from its point of view, the position of control blocks is random (as they were chosen by the sampler). It intuitively follows that the channel cannot hope to "wipe out" the short control part. At best, it can place a p fraction of errors on the control part, and it is likely that an ϵ fraction of the control blocks will be decoded correctly by $\mathrm{Dec}_{\mathrm{ctrl}}$, meaning that the correct control information s is one of the candidates s' that is considered by the decoding algorithm.

The channel C chooses the error pattern e as a function of the codeword c. However, as c looks random to the channel, the channel intuitively chooses e in a way that is independent of the seed s_{π}. Therefore, the analysis used earlier for additive channels (in which e is fixed and π is random) can be applied, and the correct message appears in the list.[7]

2.2.4 Achieving quasilinear time encoding and decoding.

In order to achieve quasilinear time encoding and decoding, we need to first verify that none of the components we use, runs in larger polynomial time. In some cases (e.g., the PRG of Forbes and Kelly [FK18]) we need to delve into the construction and analysis in order to implement it in a more efficient manner.

[7]This high level argument is an oversimplification and the actual proof is quite involved. We need to show that if the channel is able to prevent the decoding algorithm from decoding, then it can be used to break one of the pseudorandom components. A significant difficulty, is that the channel cannot run the decoding algorithm (which cannot be run by a small space ROBP) and therefore, the channel "does not know" whether it succeeded in preventing the decoding algorithm from decoding correctly. This is a problem as the distinguisher (for the PRG) that we aim to construct, will intuitively want to distinguish the output of the PRG from random, by distinguishing between the case that decoding succeeded from the one where it doesn't (and in particular, the distinguisher will want to run decoding algorithms). The argument used to construct this distinguisher relies on additional specific properties of the BSC code. Our approach to handling this issue, builds heavily on the previous arguments of [GS16, SS16] with some modifications.

We also remark that a possible behavior of a channel is to inject "false control strings" in order to make the decoding algorithm decode to incorrect values. Indeed, there are bounded space channels that can cause the decoding algorithm to have incorrect messages in the list (in addition to the correct one).

A key observation is that we don't have to optimize the exponent of the polynomial in the time of encoding and decoding $\mathrm{Enc}_{\mathrm{ctrl}}$. This is crucial as for example, the time of the decoding algorithm that we give for OddRawRS codes is inherently at least quadratic (just for writing all the large lists in the list recovering stage).

We make the following observation: When encoding, we run $\mathrm{Enc}_{\mathrm{ctrl}}$ (which in turn runs the linear code $\mathrm{Enc}_{\mathsf{OddRawRS}}$) many times. Therefore, we only care about amortized encoding time (rather than worst case running time). Any linear function $L : \mathbb{F}_2^{n^{0.1}} \to \mathbb{F}_2^n$ can be computed in amortized time $O(n \cdot (\log^2 n))$ (following a pre-processing step that prepares the matrix of L). This is because making n such computations, can be reduced to matrix multiplication of an $n \times n^{0.1}$ matrix by an $n^{0.1} \times n$ matrix (which can be done in time $O(n^2 \cdot \log^2 n)$ by Coppersmith [Cop82]. Recall that we have less than $N^{1-\lambda}$ applications of $\mathrm{Enc}_{\mathrm{ctrl}}$ where each one is over block length N^{λ}, and so overall, all these applications take time $O(N \cdot \log^2 N)$

A second observation is that when list-decoding, the decoding algorithm doesn't need to try all $n = N^{1-\lambda}$ blocks. It can instead sample a polylogarithmic number of blocks in $[n]$, and only try to decode the control code on the sampled blocks. Each such block is of length $b = N^{\lambda}$, and so, even if applying $\mathrm{Dec}_{\mathrm{ctrl}}$ takes time b^c for a large constant c, by choosing the constant $\lambda > 0$ to be sufficiently small, this step takes time $\mathrm{polylog}(N) \cdot N^{\lambda \cdot c} \leq N$.

Finally, we mention that an obvious bottleneck that prevents our encoding and decoding to run in linear time, is that computing a permutation $\pi : [N] \to [N]$ on all N inputs, requires time $\Omega(N \cdot \log N)$ just to read the inputs and write the outputs.

3 Preliminaries, and ingredients used in the construction

In this section we give formal definitions of the notions and ingredients used in the construction. We also cite previous results from coding theory and pseudorandomness that are used in the construction.

General notation. We use U_n to define the uniform distribution on n bits. The statistical distance between two distributions P, Q over Ω is $\max_{A \subseteq \Omega} |P(A) - Q(A)|$. Random variables $R_1, \ldots, R_n$ are t-wise independent if for every $i_1, \ldots, i_t \in [n]$, $R_{i_1}, \ldots, R_{i_t}$ are uniform and independent.

The Hamming weight of $x \in [q]^n$ is $\mathit{weight}(x) = |\{i : x_i \neq 0\}|$. The Hamming distance between $x, y \in [q]^n$ is $|\{i : x_i \neq y_i\}|$ and the relative Hamming distance is the Hamming distance divided by n.

3.1 Permuting strings

We will use a permutation $\pi : [n] \to [n]$ to "reorder" the bits of a string $x \in \{0,1\}^n$: The i'th bit in the rearranged string will be $\pi(i)$'th bit in x. This is captured in the definition below.

Definition 3.1 (Permuting strings). *Given a string $x \in \{0,1\}^n$ and a permutation $\pi : [n] \to [n]$. Let $\pi(x)$ denote the string $x' \in \{0,1\}^n$ with $x'_i = x_{\pi(i)}$.*

3.2 Read once branching program (in any order)

3.2.1 Formal definition of ROBPs and bounded space channels

We give a more formal definition of bounded space computation and channels, restating Definition 1.3 in a more formal notation. The model that we consider is that of oblivious read once branching programs (ROBP). In the definition below, we will consider several variants depending on whether the ROBP outputs a single bit, or one bit per any input bit (which is the case for channels that are function $C : \{0,1\}^n \to$

$\{0,1\}^n$). We will also consider ROBPs that are allowed to choose the order in which they read the input bits.

Definition 3.2 (Read Once Branching Programs (ROBP)). *A space s **ROBP** C which receives input in $\{0,1\}^n$ is defined by picking n transition functions $\delta_1,\dots,\delta_n$ where for each i, $\delta_i : \{0,1\}^s \times \{0,1\} \to \{0,1\}^s$. On input $x \in \{0,1\}^n$, the computation path of C is the sequence $r_0,\dots,r_n$ of states defined by $r_0 = 0^s$ and for $i \geq 1$, $r_i = \delta_i(r_{i-1}, x_i)$. We distinguish between two types of ROBPs:*

- *If $C : \{0,1\}^n \to \{0,1\}$ is an ROBP that outputs a single bit, then C also has an output function $o : \{0,1\}^s \to \{0,1\}$ and $C(x)$ is defined to be $o(r_n)$.*
- *If $C : \{0,1\}^n \to \{0,1\}^n$ is an ROBP that outputs n bits, then C also has n output functions $o_1,\dots,o_n$ where for each i, $o_i : \{0,1\}^s \to \{0,1\}$ and $C(x)$ is defined by the n bit string $o_1(r_1),\dots,o_n(r_n)$.*

We are stating this definition in the terminology of "transition functions" and "output functions" which is more convenient when discussing ROBPs that output more than one bit. However, we stress that this definition is equivalent to the more common definition of width $w = 2^s$ ROBPs in terms of a layered graph with $n+1$ layers, where the i'th transition function specifics the edges from the $(i-1)$'th level to the i'th level.

Another remark is that the definition above forces an ROBP that outputs many bits to output its i'th bit *before* seeing the $(i+1)$'th bit. This is done in order to have a simple definition of ROBPs that output many bits. However, all our results hold for a more general model in which the ROBP can delay outputting the i'th bit to a later stage and look ahead at the next input bits.[8]

We now define *any-order* ROBPs that are allowed to reorder their input bits using a permutation $\sigma : [n] \to [n]$ prior to reading the input. The next claim immediately follows from the definition.

Definition 3.3 (any-order ROBPs). *Given an ROBP C over n bits, and a permutation $\sigma : [n] \to [n]$ we define C^σ to be the function $C^\sigma(x) = C(\sigma(x))$. (Here $\sigma(x)$ is the function from Definition 3.1). The class of **any-order space** s **ROBPs** is the class of all functions C^σ where C is a space s ROBP and $\sigma : [n] \to [n]$ is a permutation.*

We now observe that if we restrict the input of an any-order space s ROBP, then we obtain an any-order space s ROBP.

Claim 3.4 (Restrictions of any-order ROBPs). *Given a space s ROBP $C : \{0,1\}^n \to \{0,1\}$, a permutation $\sigma : [n] \to [n]$, $T \subseteq [n]$ of size t, and $v \in \{0,1\}^t$, the function $f : \{0,1\}^{n-t} \to \{0,1\}$ defined by $f(y) = C^\sigma(x)$ where $x_T = v$ and $x_{[n]\setminus T} = y$, is computable by an any-order space s ROBP. That is, there exists a permutation $\tau : [n-t] \to [n-t]$ and a space s ROBP $D : \{0,1\}^{n-t} \to \{0,1\}$ such that $D^\tau(y) = f(y)$.*

Definition 3.3 applies also to ROBPs that output n bits. Note that in that case, the output of the ROBP is also ordered by σ. When considering channels, it is more convenient to reorder the bits back to the natural order. This is done in the next definition.

[8] To make this statement more concrete, a space s ROBP that wants to look ahead and read t additional bits before outputing the i'th output bit, can store these additional t bits, and the number of bits it outputted so far, in its memory, and this can be done in space $s + t + \log n$. Our results on space s channels also apply to these kind of channels, namely channels that use space $\Omega(s)$ and look ahead at the next $\Omega(s)$ input bits. More generally, our results apply to any "reasonable" model of ROBPs that output many bits, in which if $C_1, C_2 : \{0,1\}^n \to \{0,1\}^n$ are space s ROBPs, then the composition $C_1 \circ C_2$ can be computed by an ROBP with space, say $O(s)$.

Definition 3.5 (any-order bounded space channels)**.** *The class of* ***any-order space*** s ***channels*** *is the class of all functions* $e_C^\sigma : \{0,1\}^n \to \{0,1\}^n$*, where* $C : \{0,1\}^n \to \{0,1\}^n$ *is a space* s *ROBP,* $\sigma : [n] \to [n]$ *is a permutation, and* $e_C^\sigma(x) = \sigma^{-1}(C(\sigma(x)))$*.*

It should be noted that the "reordered function" e_C^σ is not necessarily computable by a small space ROBP. However, when applying channels e_C^σ on a codeword, it is more natural to reorder the bits in the order used by the codeword.

Using this notation, the bounded space channel model considered in [GS16, SS16] (which was called "online space s channels") corresponds to space s channels with the identity permutation (namely, channels that read their input bits in the standard order).

3.2.2 PRGs for any-order ROBPs

We need the following standard definition of pseudorandom distributions and generators.

Definition 3.6 (Pseudorandom generators)**.** *A distribution* X *on* n *bits is* ϵ*-**pseudorandom** for a class* $\mathcal{C}$ *of functions from* n *bits to one bit, if for every* $C \in \mathcal{C}$*,* $|\Pr[C(X) = 1] - \Pr[C(U_n)] = 1| \leq \epsilon$*. A function* $G : \{0,1\}^d \to \{0,1\}^n$ *is an* ϵ*-**PRG** for* $\mathcal{C}$ *if* $G(U_d)$ *is* ϵ*-pseudorandom for* $\mathcal{C}$*.*

We will use the following PRG by Forbes and Kelly [FK18]

Theorem 3.7. *[FK18] For every* $\log n \leq s \leq n$*, there exists an* ϵ*-PRG* $G : \{0,1\}^d \to \{0,1\}^n$ *for any-order space* s *ROBPs that output one bit, with* $d = O((s + \log \frac{1}{\epsilon}) \cdot \log^2 n)$*. Furthermore,* G *can be computed in time* $O(n \cdot \mathit{polylog}(n))$*.*

Forbes and Kelly [FK18] do not carefully estimate the running time of their pseudorandom generator, and only claim that it runs in polynomial time. The proof below, will prove the "furthermore" clause in the theorem above.

Proof. (of the "furthermore" clause in Theorem 3.7) The construction of Forbes and Kelly works as follows: Let k be a parameter to be chosen later. The generator G is constructed iteratively, by setting G_0 to be a $320k$-wise independent distribution, and $G_{i+1} = D_i \oplus (T_i \wedge G_i)$ where D_i is a $2k$-wise independent distribution, and T_i is a k-wise independent distribution. (Different copies of D_i's and T_i's are sampled independently). The final distribution is $G = G_r$ where r is a parameter chosen by the construction.

Note that sampling a k-wise independent distribution X on n bits can be done by a deterministic procedure that receives a seed of length $k \log n$, in time $n \cdot \mathrm{polylog}(n)$. This can be done by the standard Reed-Solomon based construction. Namely, encoding the $k \log n$ bit seed as k elements $a_0, \ldots, a_{k-1} \in \mathbb{F}_n$ (here we assume w.l.o.g. that n is a power of 2) and for $\alpha \in [n]$ (which can be interpreted as $\alpha \in \mathbb{F}_n$) setting X_α to be (the first bit of) $\sum_{0 \leq i < k} a_i \alpha^i$. This gives a k-wise independent distribution and can be computed in time $n \cdot \mathrm{polylog}(n)$, using univariate multipoint evaluation, that can be done with $O(n \log n)$ field operations.

Forbes and Kelly show that taking $r = O(\log n)$ and $k = O(s + \log n + \log(1/\epsilon))$ gives an ϵ-PRG for any-order space s ROBPs. This gives total running time of $r \cdot n \cdot \mathrm{polylog}(n) = n \cdot \mathrm{polylog}(n)$.

This follows by the proof of [FK18, Lemma 4.2] which proves the correctness of the PRG with a slightly different construction, but also applies to the construction described above. ☐

3.3 Averaging Samplers

The reader is referred to Goldreich's survey [Gol97] on averaging samplers.

Definition 3.8 (Averaging Samplers). *A function* $\mathrm{Samp} : \{0,1\}^n \to (\{0,1\}^m)^t$ *is an* (ϵ, δ)-***Sampler*** *if for every* $f : \{0,1\}^m \to [0,1]$, $\Pr_{(z_1,\ldots,z_t)\leftarrow \mathrm{Samp}(U_n)}[|\frac{1}{t}\sum_{i\in[t]} f(z_i) - \frac{1}{2^m}\sum_{x\in\{0,1\}^m} f(x)| > \epsilon] \leq \delta$. *A sampler has* ***distinct samples*** *if for every* $x \in \{0,1\}^n$*, the* t *elements in* $\mathrm{Samp}(x)$ *are distinct.*

The next theorem follows from the "expander sampler". This particular form can be found (for example) in [Vad04].

Theorem 3.9. *For every sufficiently large* m *and every* $\epsilon \geq \delta > 0$ *such that* $m \leq \log(1/\delta)$ *there is an* (ϵ, δ)*-sampler with distinct samples,* $\mathrm{Samp} : \{0,1\}^{O(\log(1/\delta)\cdot \mathrm{poly}(1/\epsilon))} \to (\{0,1\}^m)^t$ *for any* $t \leq 2^m$ *such that* $t \geq \mathrm{poly}(1/\epsilon) \cdot \log(1/\delta)$*. Furthermore,* Samp *is computable in time* $t \cdot \mathrm{poly}(1/\epsilon, \log(1/\delta))$ *and has distinct samples.*

3.4 Almost t-wise permutations

We also need the following notion of almost t-wise permutations.

Definition 3.10 (Almost t-wise independent permutations). *A function* $\pi : \{0,1\}^d \times [n] \to [n]$ *is an* (ϵ, t)*-wise independent permutation if:*

- *For every* $s \in \{0,1\}^d$ *the function* $\pi_s(i) = \pi(s,i)$ *is a permutation over* $[n]$.
- *For every distinct* $i_1, \ldots, i_t \in [n]$*, the random variable* $R = (R_1, \ldots, R_t)$ *defined by* $R_j = \pi(s, i_j) \colon s \leftarrow U_d$*, is* ϵ*-close to* t *uniform samples without repetition from* $[n]$.

Theorem 3.11. *[KNR09] For every* t *and every sufficiently large* n*, there exists an* (ϵ, t)*-wise independent permutation with* $d = O(t \cdot \log n + \log(1/\epsilon))$*. Furthermore, computing* $\pi(s, i)$ *on inputs* $s \in \{0,1\}^d$ *and* $i \in [n]$ *can be done in time* $\mathrm{poly}(d, \log n)$.[9]

We will use (ϵ, t)-wise independent permutations to permute strings. Consider the following example: Let $e \in \{0,1\}^n$ be a string with Hamming weight pn, and let $\pi : \{0,1\}^d \times [n] \to [n]$ be an (ϵ, t)-wise independent permutation. We will be interested in the distribution $X = \pi_{U_d}(e)$ (here, $\pi(e)$ is the "permuted string" defined in Definition 3.1). We would like to apply "Chernoff style bounds for t-wise independence" [BR94, SSS95] on $X_1, \ldots, X_n$. A technical issue is that it is not true that $X_1, \ldots, X_n$ are t-wise independent (even in the case that $\epsilon = 0$). What is true is that for every t-tuple of distinct indices $i_1, \ldots, i_t \in [n]$, $\Pr[X_{i_1} = \ldots = X_{i_t} = 1] \leq p^t + \epsilon$. The latter condition is sufficient to obtain Chernoff style behavior (at least when ϵ is sufficiently small compared to p^t) by the following lemma.

Lemma 3.12 (tail bounds for almost t-wise independent permuted strings). *Let* $X_1, ..., X_n$ *be binary random variables, such that for every set of distinct* t *indices* $i_1, \cdots, i_t \in [n]$, $\Pr[X_{i_1} = \ldots = X_{i_t} = 1] \leq \mu^t$*. If* $0 < \delta \leq 1$ *and* $t \leq \frac{\delta \cdot \mu \cdot n}{2}$ *then*

$$Pr[\sum_{j=1}^{n} X_j \geq (1+\delta) \cdot \mu \cdot n] \leq e^{-\Omega(\delta t)}$$

[9]We will be interested in the time it takes to compute the permutation on all $i \in [n]$ (namely given s, we want to compute $(\pi(s,i))_{i\in[n]}$) and will use $n \cdot \mathrm{poly}(d)$ as a bound on the time for this task. Note that this also gives that computing $(\pi^{-1}(s,i))_{i\in[n]}$ can be done within the same time bound.

This type of lemma follows by using the approach of [SSS95]. It was applied for t-wise independent permutations (in a related setup) in [DHRS07]. For completeness we provide a proof.

Proof. (of Lemma 3.12) We use X to denote the n bit long random variable composed of $(X_1, \ldots, X_n)$. For every t-tuple $y = (i_1, \ldots, i_t)$ of indices in $[n]$, let A_y be the event $A_y = \{X_{i_1} = \ldots, X_{i_t} = 1\}$. Let Y be an independent, uniformly chosen t-tuple of indices in $[n]$. We have that $\Pr[A_Y] \le \mu^t$. Let $\ell = (1+\delta) \cdot \mu \cdot n$. Note that for every $x \in \{0,1\}^n$ such that $\sum_{j \in [n]} x_j \ge \ell$, we have that:

$$\Pr[A_Y | X = x] \ge \frac{\binom{\ell}{t}}{\binom{n}{t}} = \frac{\ell \cdot \ldots \cdot (\ell - t + 1)}{n \cdot \ldots \cdot (n - t + 1)} \ge \frac{(\ell - t + 1)^t}{n^t} \ge \mu^t \cdot (1 + \frac{\delta}{2})^t.$$

This gives that,

$$\Pr[A_Y] \ge \Pr[A_Y \cap \{X \ge \ell\}] \ge \Pr[X \ge \ell] \cdot \Pr[A_Y | X \ge \ell] \ge \Pr[X \ge \ell] \cdot \mu^t \cdot (1 + \frac{\delta}{2})^t.$$

Rearranging, we get that:

$$\Pr[X \ge \ell] \le \frac{\Pr[A_Y]}{\mu^t \cdot (1 + \frac{\delta}{2})^t} \le \frac{\mu^t}{\mu^t \cdot (1 + \frac{\delta}{2})^t} = e^{-\Omega(\delta \cdot t)}.$$

□

3.5 Error-Correcting Codes

In this section we give definitions of the various notions of error correcting codes used in this paper. We also state some previous constructions that will be used in this paper.

3.5.1 The standard notion of error correcting codes

We give a more general version of Definition 1.1 that discusses codes over non-binary alphabets, as well as codes in Shannon's scenario. For our purposes it is more natural to define codes in terms of a pair $(\mathrm{Enc}, \mathrm{Dec})$ of encoding and decoding algorithms. Different variants are obtained by considering different properties required by the encoding and decoding algorithms and different types of error patterns.

Definition 3.13 (Codes). *Let k, n, q be parameters and let $\mathrm{Enc} : \{0,1\}^k \to (\{0,1\}^{\log q})^n$ be a function. We say that Enc is an encoding function for a code that is:*

- ***decodable from t errors**, if $t \in [n]$, and there exists a function $\mathrm{Dec} : (\{0,1\}^{\log q})^n \to \{0,1\}^k$ such that for every $m \in \{0,1\}^k$ and every $e \in (\{0,1\}^{\log q})^n$ with Hamming weight at most t, $\mathrm{Dec}(\mathrm{Enc}(m) \oplus e) = m$.*
- *L-**list-decodable from t errors**, if the function Dec is allowed to output a list of size at most L, and for every $m \in \{0,1\}^k$ and every $e \in (\{0,1\}^{\log q})^n$ with Hamming weight at most t, $\mathrm{Dec}(\mathrm{Enc}(m) \oplus e) \ni m$.*
- ***decodable from** P, with success probability $1 - \nu$, if P is a distribution over $(\{0,1\}^{\log q})^n$, $0 \le \nu \le 1$, and there exists a function $\mathrm{Dec} : (\{0,1\}^{\log q})^n \to \{0,1\}^k$ such that for every $m \in \{0,1\}^k$, $\Pr_{e \leftarrow P}[\mathrm{Dec}(\mathrm{Enc}(m) \oplus e) = m] \ge 1 - \nu$.*

A code has ***encoding time*** *[resp.* ***decoding time****]* $T(\cdot)$*, if* Enc *[resp.* Dec*] can be computed in time* $T(n \log q)$*. The code is* ***explicit*** *if both encoding and decoding run in polynomial time. (Naturally, this makes sense only for a family of encoding and decoding functions with varying block length* n*, message length* $k(n)$*, and alphabet size* $q(n)$*).*

The rate of the code is the ratio of the message length and output length of Enc*, where both lengths are measured in bits. That is the rate* $R = \frac{k}{n \cdot \log q}$*.*

3.5.2 Linear Codes and Dual Distance

We also define the standard notion of linear codes.

Definition 3.14 (Linear codes and dual codes)**.** *Let* q *be a prime power, and let* $\mathbb{F}_q$ *denote the field with* q *element. An* $[n,k]_q$ ***linear code*** *is a linear subspace of* $C \subseteq \mathbb{F}_q^n$ *of dimension* k*. We say that* C *has distance* d*, if the Hamming weight of every nonzero vector in* C *is at least* d*. Such codes are called* $[n,k,d]_q$ *codes. A linear map* $\mathrm{Enc} : \mathbb{F}_q^k \to \mathbb{F}_q^n$ *is an* ***encoding function*** *for* C*, if* $\mathrm{Enc}(\mathbb{F}_q^k) = C$*. For a code* C*, we use* $C^{\perp}$ *to denote the dual vector space. We say that* C *has* ***dual distance*** d *if* $C^{\perp}$ *has distance* d*.*

It is standard that C is an $[n,k,2t+1]_q$ code iff C has a linear encoding function $\mathrm{Enc} : \mathbb{F}_q^k \to \mathbb{F}_q^n$ that is decodable from t errors. We will use the standard fact that encoding functions for codes with dual distance r yield $(r-1)$-wise dual independent distributions.

Lemma 3.15 ($(t-1)$-wise independence from linear codes with dual distance t)**.** *Let* $\mathrm{Enc} : \mathbb{F}_q^k \to \mathbb{F}_q^n$ *be an encoding function for a linear* $[n,k]_q$*-code* C *with dual distance* t*. Applying* Enc *on a uniformly chosen message* $m \leftarrow \mathbb{F}_q^k$ *yields a distribution* $(Z_1, \ldots, Z_n)$ *over* $\mathbb{F}_q^n$ *that is* $(t-1)$*-wise independent, and every* Z_i *is uniformly distributed over* $\mathbb{F}_q$*.*

Proof. Applying Enc on some $v \in \mathbb{F}^k$, can be seen as multiplying v by a generator matrix of C (which is a transposed parity check matrix of $C^{\perp}$). As $C^{\perp}$ has distance t, every $t-1$ columns of the generator matrix of C are linearly independent, and the lemma follows. $\square$

3.5.3 Stochastic Codes

We restate Definition 1.2 using slightly more precise notation.

Definition 3.16 (Stochastic codes for channels)**.** *Let* k, n, d *be parameters and let* $\mathrm{Enc} : \{0,1\}^k \times \{0,1\}^d \to \{0,1\}^n$ *be a function. Let* $\mathcal{C}$ *be a class of functions from* n *bits to* n *bits. We say that* Enc *is an encoding function for a stochastic code that is:*

- ***decodable*** *for "channel class"* $\mathcal{C}$*, with success probability* $1-\nu$*, if there exists a (possibly randomized) procedure* $\mathrm{Dec} : \{0,1\}^n \to \{0,1\}^k$ *such that for every* $m \in \{0,1\}^k$ *and every* $C \in \mathcal{C}$*, setting* $X = \mathrm{Enc}(m, U_d)$*, we have that* $\Pr[\mathrm{Dec}(X \oplus C(X)) = m] \geq 1 - \nu$*, where the probability is over coin tosses of the encoding and decoding procedures.*
- L*-****list-decodable*** *for "channel class"* $\mathcal{C}$*, with success probability* $1-\nu$*, if the procedure* Dec *is allowed to output a list of size at most* L*, and* $\Pr[\mathrm{Dec}(X \oplus C(X)) \ni m] \geq 1 - \nu$*, where the probability is over coin tosses of the encoding and decoding procedures.*

A code has ***encoding time*** *[resp.* ***decoding time****]* $T(\cdot)$*, if* Enc *[resp.* Dec*] can be computed in time* $T(k + n + d)$*. The code is* ***explicit*** *if both encoding and decoding run in polynomial time. (Naturally, this makes* *sense only for a family of encoding and decoding functions with varying block length n, message length $k(n)$ and seed length $d(n)$).*

The rate of the code is the ratio of the message length and output length of Enc*, where both lengths are measured in bits. That is the rate $R = \frac{k}{n}$.*

3.5.4 Codes for binary-symmetric channels and related variants

We will make use of known constructions of codes for binary symmetric channels.

Definition 3.17 (Binary symmetric channel)**.** *Let BSC_p^n denote the distribution over n bit strings in which individual bits are i.i.d. and each is one with probability p.*

There are constructions of codes with rate approaching $1 - H(p)$ that are decodable from BSC_p^n with very high suvcess probability, and have linear time encoding and decoding [GI05].

We are interested in codes for an intuitively similar (but more general) scenario in which the error distribution is obtained by taking a string $e \in \{0,1\}^n$ of weight pn, an (ϵ, t)-wise independent permutation $\pi : \{0,1\}^d \times [n] \to [n]$ and considering the error distribution $e' = \pi_{U_d}(e)$.

This distribution is somewhat similar to BSC_p in the sense that if we project both distributions to a "not too large" tuple of indices, the distributions are statistically close. More precisely, for every choice of t distinct indices $I = (i_1, \ldots, i_t)$, the distribution $(\mathrm{BSC}_p)_I$ and $(\pi_{U_d}(e))_I$ are $(\epsilon + t^2/n)$-close in statistical distance. This can be used to argue that current constructions for BSC_p also work for $\pi_{U_d}(e)$ (for certain parameters).

However, for technical reasons, this isn't sufficient for our purposes, and we will require that the code has some additional structure (which we will use in our construction). We now explain the additional structure that we need: The known codes for BSC_p are constructed by code concatenation, and for technical reasons, we will be interested in some properties of the inner and outer codes (and not just properties of the concatenated code). We first give the following standard definition of code concatenation.

Definition 3.18 (Concatenated code)**.** *Given functions:*

- $\mathrm{Enc}_{\mathrm{out}} : \{0,1\}^{k_{\mathrm{out}}} \to (\{0,1\}^{\log q_{\mathrm{out}}})^{n_{\mathrm{out}}}$*, and*
- $\mathrm{Enc}_{\mathrm{in}} : \{0,1\}^{k_{\mathrm{in}}} \to (\{0,1\}^{\log q_{\mathrm{in}}})^{n_{\mathrm{in}}}$,

*such that $\log q_{\mathrm{out}} = k_{\mathrm{in}}$ we define the **concatenated encoding function** $\mathrm{Enc} : \{0,1\}^{k_{\mathrm{out}}} \to (\{0,1\}^{\log q_{\mathrm{in}}})^{n_{\mathrm{out}} \cdot n_{\mathrm{in}}}$ denoted by $\mathrm{Enc}_{\mathrm{out}} \circ \mathrm{Enc}_{\mathrm{in}}$ as follows: For $i_{\mathrm{out}} \in [n_{\mathrm{out}}]$, $i_{\mathrm{in}} \in [n_{\mathrm{in}}]$, and $i = (i_{\mathrm{out}} - 1) \cdot n_{\mathrm{in}} + i_{\mathrm{in}}$ we define $\mathrm{Enc}(m)_i = \mathrm{Enc}_{\mathrm{in}}(\mathrm{Enc}_{\mathrm{out}}(m)_{i_{\mathrm{out}}})_{i_{\mathrm{in}}}$.*

Concatenated codes can be decoded by "concatenated decoding".

Definition 3.19 (Concatenated decoding)**.** *Let $\mathrm{Enc} = \mathrm{Enc}_{\mathrm{out}} \circ \mathrm{Enc}_{\mathrm{in}}$ be a concatenated code, and let $\mathrm{Dec}_{\mathrm{out}} : (\{0,1\}^{\log q_{\mathrm{out}}})^{n_{\mathrm{out}}} \to \{0,1\}^{k_{\mathrm{out}}}$, $\mathrm{Dec}_{\mathrm{in}} : (\{0,1\}^{\log q_{\mathrm{in}}})^{n_{\mathrm{in}}} \to \{0,1\}^{k_{\mathrm{in}}}$ be functions. For $i \in [n_{\mathrm{out}}]$ we define $\mathrm{Dec}_{\mathrm{in}}^i : (\{0,1\}^{\log q_{\mathrm{in}}})^{n_{\mathrm{out}} \cdot n_{\mathrm{in}}} \to \{0,1\}^{k_{\mathrm{in}}}$ by:*

$$\mathrm{Dec}_{\mathrm{in}}^i(z) = (z_{(i-1) \cdot n_{\mathrm{in}} + 1}, \ldots, z_{i \cdot n_{\mathrm{in}}}).$$

*The **concatenated decoding** function $\mathrm{Dec} : (\{0,1\}^{\log q_{\mathrm{in}}})^{n_{\mathrm{out}} \cdot n_{\mathrm{in}}} \to \{0,1\}^{k_{\mathrm{out}}}$ is defined by:*

$$\mathrm{Dec}(z) = \mathrm{Dec}_{\mathrm{out}}(\mathrm{Dec}_{\mathrm{in}}^1(z), \ldots, \mathrm{Dec}_{\mathrm{in}}^{n_{\mathrm{out}}}(z)).$$

In the following theorem we revisit the code construction of [GI05] for BSC_p^n, and observe that the constructed concatenated code has some properties that we will use later on.

Theorem 3.20. *For every constant $0 < p < 1/2$, and every sufficiently small constant $\epsilon > 0$, there exist integer constants $k_{\mathrm{in}}, n_{\mathrm{in}}, q_{\mathrm{out}}$ and real constants $\lambda_1, \lambda_2, \lambda_3 > 0$ such that $k_{\mathrm{in}} = \log q_{\mathrm{out}} \leq \frac{1}{\epsilon^7}$, and for infinitely many choices of n_{out} there exist functions:*

- $\mathrm{Enc}_{\mathrm{out}} : \{0,1\}^{k_{\mathrm{out}}} \to (\{0,1\}^{\log q_{\mathrm{out}}})^{n_{\mathrm{out}}}$,
- $\mathrm{Enc}_{\mathrm{in}} : \{0,1\}^{k_{\mathrm{in}}} \to \{0,1\}^{n_{\mathrm{in}}}$,

such that:

- *$R_{\mathrm{out}} = \frac{k_{\mathrm{out}}}{n_{\mathrm{out}} \cdot \log q_{\mathrm{out}}} \geq 1 - \frac{\epsilon}{10}$, and $\mathrm{Enc}_{\mathrm{out}}$ is decodable from $w = \lambda_1 \cdot n_{\mathrm{out}}$ errors with linear time encoding and decoding.*
- *$R_{\mathrm{in}} = \frac{k_{\mathrm{in}}}{n_{\mathrm{in}}} \geq 1 - H(p) - \epsilon/10$, and $\mathrm{Enc}_{\mathrm{in}}$ is decodable from $\mathrm{BSC}_p^{n_{\mathrm{in}}}$ with probability $1 - 2^{-\lambda_2 \cdot n_{\mathrm{in}}}$. This decoding is achieved by a function $\mathrm{Dec}_{\mathrm{in}}$ that implements "maximum likelihood decoding".*
- *Consequently, setting $n = n_{\mathrm{out}} \cdot n_{\mathrm{in}}$, and $q_{\mathrm{in}} = 2$, the concatenated code $\mathrm{Enc} = \mathrm{Enc}_{\mathrm{out}} \circ \mathrm{Enc}_{\mathrm{in}} : \{0,1\}^{k_{\mathrm{out}}} \to \{0,1\}^n$ is well defined, has rate $R = \frac{k_{\mathrm{out}}}{n} \geq 1 - H(p) - \epsilon$, and is encodable in time $O(n)$ (where the constant c hidden in the $O(\cdot)$ depends on ϵ, and $c = c(\epsilon) = 2^{\mathrm{poly}(1/\epsilon)}$).*
- *Let $t \leq n^{0.1}$, and let $\pi : \{0,1\}^d \times [n] \to [n]$ be a $(2^{-10 \cdot t}, t)$-wise independent permutation. Let $m \in \{0,1\}^{k_{\mathrm{out}}}$, and let $A_m : \{0,1\}^n \to \{0,1\}$ be the function that on input $e' \in \{0,1\}^n$, outputs one iff*
  $$|\left\{i \in [n_{\mathrm{out}}] : \mathrm{Dec}_{\mathrm{in}}^i(\mathrm{Enc}(m) \oplus e') \neq \mathrm{Enc}_{\mathrm{out}}(m)_i\right\}| \leq \frac{w}{10}.$$
  (Note that $A_m(e') = 1$ implies that concatenated decoding that is applied on $\mathrm{Enc}(m) \oplus e'$ indeed recovers m).

  For every $e \in \{0,1\}^n$ of Hamming weight at most pn,
  $$\Pr[A_m(\pi_{U_d}(e)) = 1] \geq 1 - 2^{-\lambda_3 \cdot t}.$$
- *Consequently, for every $e \in \{0,1\}^n$ of Hamming weight at most pn, the code Enc is decodable from $\pi_{U_d}(e)$ with probability $1 - 2^{-\lambda_3 \cdot t}$. Furthermore, the concatenated decoding algorithm runs in time $O(n)$ (where the constant c hidden in the $O(\cdot)$ depends on ϵ, and $c = c(\epsilon) = 2^{\mathrm{poly}(1/\epsilon)}$).*

The final item in Theorem 3.20 follows from the penultimate item. However, for our purposes, the final item will not be sufficiently strong, and we will need to use the penultimate item (as well as the previous items). The advantage of the penultimate item is that we get that for every m, there exists a space $O(\log n)$ ROBP which implements the function A_m, in contrast to the entire concatenated decoding algorithm that does not seem to be implemented by small space ROBPs.

Theorem 3.20 follows by noticing that the proofs of known construction of codes for binary symmetric channels (see e.g., [For65]) are achieved by code concatenation of codes with the properties listed above. The fourth item follows by using Lemma 3.12 to analyze the behavior of this concatenated code on errors from the distribution $\pi_{U_d}(e)$. The proof appears in Appendix A.

Very similar arguments to the proof of Theorem 3.20 were made by Smith [Smi07] and in an early version of [GS16].

4 Raw Reed-Solomon Codes

For our intended application, we need linear binary $[n, k]_2$ codes with:

- Relative distance $(1/2 - o(1))$.
- Large dual distance of at least $n^{\Omega(1)}$. (In fact, we need a slightly stronger property to be explained below).
- Polynomial time encoding.
- Polynomial time unique decoding from p-fraction errors for every $p < \frac{1}{4}$.
- Polynomial time list-decoding with list size $\mathrm{poly}(1/\epsilon)$ from $(\frac{1}{2} - \epsilon)$-fraction errors, for every constant $\epsilon > 0$.

In this paper, we construct binary codes with the properties above. To the best of our knowledge, this is the first construction of such codes.

First a slight abuse of notation: for this section only, we will use the word distance to denote *relative distance*, as opposed to *absolute distance*. This helps with the exposition.

Reed Solomon codes exhibit all the properties above (in addition to constant rate, and larger dual distance) but only for large alphabets. As far as we are aware, there are only two known families of codes over the binary alphabet which have $\Omega(1)$ distance and $n^{\Omega(1)}$ absolute dual distance. The first family is dual-BCH codes, but we do not know decoding algorithms for these codes from $\Omega(1)$-fraction errors for this setting of parameters (it is known [KS07, KS13] how to decode from $\Omega(1)$-fraction errors only when the absolute dual distance is $O(\log n)$). The second family is based on Algebraic-Geometric codes (see the appendix to [Shp09] for a detailed exposition). AG codes are generalizations of Reed-Solomon codes, and retain many of the good features of Reed-Solomon codes while having the advantage of being realizable over constant size alphabets. An AG code with suitable parameters over a constant size alphabet $\mathbb{F}_{2^t}$ has $\Omega(1)$ distance and $\Omega(1)$ dual distance. To bring the alphabet down to binary, one can do code concatenation. However, typically *concatenation destroys dual distance*. But not always! If we concatenate with a *trivial code*,[10] that maps $\mathbb{F}_{2^t}$ to t-bit strings, the absolute dual distance is preserved under concatenation. On the other hand, using the trivial code makes the distance shrink by a factor t. This yields codes with $\Omega(1)$ distance and dual distance with efficient decoding algorithms (these are the codes that we use to prove Theorem 1.5). However the lower bound on the distance that follows is nowhere near[11] $1/2$.

The binary codes that we construct here are obtained by concatenating Reed-Solomon codes (over a large alphabet) with a different trivial code for each coordinate of the Reed-Solomon code. We call the general class of such codes *Raw Reed-Solomon codes*. In the positive direction, concatenating with trivial codes preserves the absolute dual distance, and we get the required dual-distance property. On the other hand, since the outer Reed-Solomon code is over a large alphabet, the trivial codes must have superconstant block-length, and thus $o(1)$ distance. By default, concatenating with inner codes of $o(1)$ distance leads to the final codes having $o(1)$ distance ($O(1/\log n)$ to be precise). However, for *special choices* of the trivial codes, we use some deep algebraic tools[12] to give a direct analysis of the distance of these codes, which miraculously turns out to be $1/2 - o(1)$. Finally, using the powerful list-decoding machinery available for

[10] In this paper, "trivial code" will always refer to bijective $\mathbb{F}_2$-linear maps $\Phi : \mathbb{F}_{2^m} \to \mathbb{F}_2^m$ for some m. They are trivial because their absolute minimum distance equals 1. Note that for any given m, there are many different choices of trivial codes (corresponding to invertible $m \times m$ matrices over $\mathbb{F}_2$).

[11] The lower bound obtained on the distance of the resulting codes is always at most $\frac{5}{84} < 0.06$.

[12] The Weil bounds, which are also used to analyze the distance of dual-BCH codes.

Reed-Solomon codes, we show that Raw Reed-Solomon codes can be list-decoded from nearly $(1/2-\epsilon)$-fraction errors.

Reed Solomon codes have a lot of structure and many useful properties (in addition to their distance properties) and so, we believe that the fact that Reed-Solomon codes have the additional properties listed above (when viewed as binary codes appropriately) is of independent interest and may prove useful in other applications.

In the theorem below, we focus on codes with the parameters that we require for our application. The theorem below will follow from a more general result on Raw Reed-Solomon codes stated later.

Theorem 4.1 (Codes with large distance and dual distance)**.** *For every constant* $0<\alpha<1/2$*, and every sufficiently large* m*, setting* $n=(2^m-1)\cdot m$*, and* $k=n^\alpha$*, there is a binary linear* $[n,k]_2$*-code* C *that satisfies:*

- C *has a linear encoding map* $\mathrm{Enc}:\mathbb{F}_2^k\to\mathbb{F}_2^n$ *that runs in time* $\mathrm{poly}(n)$*.*
- C *has relative distance* $(\frac{1}{2}-O((\frac{\log n}{n})^{\frac{1}{2}-\alpha}))=(\frac{1}{2}-o(1))$*.*
- *For every constant* $p<1/4$*,* Enc *is decodable from* p*-fraction errors in time* $\mathrm{poly}(n)$*.*
- *There exists a universal constant* b *such that for every* $\epsilon\geq b\sqrt{\alpha}$*,* Enc *is* $O(\frac{1}{\epsilon^2})$*-list-decodable from* $(\frac{1}{2}-\epsilon)$*-fraction errors in time* $\mathrm{poly}(n)$*.*[13]
- C *has absolute dual distance* $\Omega(\frac{n^\alpha}{\log n})$*.*
- *Moreover, define* $\mathrm{Enc}^{\mathrm{trunc}}:\mathbb{F}_2^{k/2}\to\mathbb{F}_2^n$ *by* $\mathrm{Enc}^{\mathrm{trunc}}(x)=\mathrm{Enc}(x\circ 0^{k/2})$*, and consider the linear code* $C'=\mathrm{Enc}^{\mathrm{trunc}}(\mathbb{F}_2^{k/2})$*. It holds that* C' *has absolute dual distance* $\Omega(\frac{n^\alpha}{\log n})$*.*

In the remainder of the section we introduce Raw Reed-Solomon codes, study their properties, and use them to prove Theorem 4.1.

4.1 General Raw Reed-Solomon codes

Let $q=2^m$. We will discuss a family of binary codes that are derived from Reed-Solomon codes over $\mathbb{F}_q$.

Start with an evaluation domain $D\subseteq\mathbb{F}_q$ and a degree bound d, and consider the Reed-Solomon code of evaluations on D of polynomials of degree at most d over $\mathbb{F}_q$. In order to convert this code to a binary code, we also choose a sequence $\mathbf{\Phi}=(\Phi_x)_{x\in D}$, where each Φ_x is an $\mathbb{F}_2$-linear bijection between $\mathbb{F}_q$ and $\mathbb{F}_2^m$.

In terms of this data, we define the Raw Reed-Solomon code $\mathsf{RawRS}[\mathbb{F}_q,d,D,\mathbf{\Phi}]$ as follows. The coordinates of the code are indexed by pairs $(x,i)\in D\times[m]$, and the codewords are indexed by polynomials $P(X)\in\mathbb{F}_q[X]$ of degree at most d. The codeword corresponding to $P(X)$ is $c:D\times[m]\to\mathbb{F}_2$ given by:

$$c(x,i)=\Phi_x(P(x))_i.$$

This can also be expressed as the Reed-Solomon code concatenated with a different trivial code $\Phi_x:\mathbb{F}_q\to\mathbb{F}_2^m$ in each coordinate (in the spirit of Justesen [Jus72] and Thommesen [Tho83]).

A lot, but not all, of requirements for the code we desire are already satisfied by arbitrary Raw Reed-Solomon codes. We now pick out two special codes in this family, $\mathsf{SimpleRawRS}$ and $\mathsf{OddRawRS}$ which do satisfy all the requirements (and whose analysis will be more specialized).

[13] We remark that the constant hidden in the notation $\mathrm{poly}(n)$ here (and in the previous item) is universal and does not depend on α. However, the choice of which m is sufficiently large, does depend on α.

- $\mathsf{SimpleRawRS}$: Let Φ be an $\mathbb{F}_2$-linear bijection from $\mathbb{F}_q$ to $\mathbb{F}_2^m$. Let $D = \mathbb{F}_q \setminus \{0\}$. For each $x \in D$, define[14] $\Phi_x(y) = \Phi(xy)$ for all y, and take $\mathbf{\Phi} = (\Phi_x)_{x \in D}$. The Simple Raw Reed-Solomon code $\mathsf{SimpleRawRS}[\mathbb{F}_q, d, \Phi]$ is defined to be $\mathsf{RawRS}[\mathbb{F}_q, d, D, \mathbf{\Phi}]$.

  In this code, the codeword corresponding to polynomial $P(X)$ is obtained by writing down, for each $x \in D$, the m bits of $\Phi(xP(x))$. Observe that $XP(X)$ is a polynomial of degree at most $d+1$ with 0 constant term. Thus, the codewords are obtained by taking a polynomial of degree at most $d+1$ with 0 constant term and writing down all its values using Φ. This is the way these codes are described in the introduction.

- $\mathsf{OddRawRS}$: Let Φ be an $\mathbb{F}_2$-linear bijection from $\mathbb{F}_q$ to $\mathbb{F}_2^m$. Let $D = \mathbb{F}_q \setminus \{0\}$. For each $x \in D$, define[15] $\Phi_x(y) = \Phi(xy^2)$ for all y, and take $\mathbf{\Phi} = (\Phi_x)_{x \in D}$. The Odd Raw Reed-Solomon code $\mathsf{OddRawRS}[\mathbb{F}_q, d, \Phi]$ is defined to be $\mathsf{RawRS}[\mathbb{F}_q, d, D, \mathbf{\Phi}]$.

  In this code, the codeword corresponding to polynomial $P(X)$ is obtained by writing down, for each $x \in D$, the m bits of $\Phi(xP(x)^2)$. Observe that $XP(X)^2$ is a polynomial of degree at most $2d+1$ with only odd degree monomials. Thus, the codewords are obtained by taking a polynomial of degree at most $2d+1$ with only odd degree monomials and writing down all its values using Φ. This is the way these codes are described in the introduction.

Our results for $\mathsf{OddRawRS}$ are technically simpler and quantitatively stronger, but $\mathsf{SimpleRawRS}$ is arguably a more natural code whose parameters are not far behind, so we feel it is interesting to see that too.

For contrast, it is also worth keeping in mind the following example:

- $\mathsf{VerySimpleRawRS}$: Let Φ be an $\mathbb{F}_2$-linear bijection from $\mathbb{F}_q$ to $\mathbb{F}_2^m$. Let $D = \mathbb{F}_q \setminus \{0\}$. For each $x \in D$, define $\Phi_x = \Phi$, and take $\mathbf{\Phi} = (\Phi_x)_{x \in D}$. The Very Simple Raw Reed-Solomon code $\mathsf{VerySimpleRawRS}[\mathbb{F}_q, d, \Phi]$ is defined to be $\mathsf{RawRS}[\mathbb{F}_q, d, D, \mathbf{\Phi}]$.

  In this code, the codeword corresponding to polynomial $P(X)$ is obtained by writing down, for each $x \in D$, the m bits of $\Phi(P(x))$. Thus, the codewords are obtained by taking a polynomial of degree at most d and writing down all its values using Φ.

  Note that this code is the usual concatenation of Reed-Solomon codes with the trivial code given by the map Φ. Also note that this code is very closely related to $\mathsf{SimpleRawRS}$: it is obtained by adding the constant functions to a suitable $\mathsf{SimpleRawRS}$.

Our plan now is as follows. First we study some properties of all Raw Reed-Solomon codes, including the rate and dual-distance. Next we prove the list-decodability of all Raw Reed-Solomon codes: this is more sophisticated, but still works in full generality.

Finally, we give a specialized analysis to show that $\mathsf{SimpleRawRS}$ and $\mathsf{OddRawRS}$ have good distance (nearly $1/2$ for the setting of interest). This is in contrast to $\mathsf{VerySimpleRawRS}$ which has distance $O(1/\log n)$.

Lemma 4.2 (Easy Properties of Raw Reed-Solomon codes). *Let $\mathbb{F}_q, d, D, m, \mathbf{\Phi}$ be as above, and let $C = \mathsf{RawRS}[\mathbb{F}_q, d, D, \mathbf{\Phi}]$. Then:*

1. *The block-length of C is $m \cdot |D|$.*
2. *The dimension of C is $m \cdot (d+1)$.*

[14]Note that for $x \in D$, $y \mapsto xy$ is a linear bijection of $\mathbb{F}_q$.

[15]Note that $y \mapsto y^2$ is a linear bijection of $\mathbb{F}_q$.

3. *C has absolute dual distance at least $(d+2)$.*

Proof. The first two items are trivial.

To show the final item, we use an alternate characterization of the dual distance: a linear code has absolute dual distance at least b if and only if the uniform distribution on the codewords of the code is $(b-1)$-wise independent.

Let $P(X) \in \mathbb{F}_q[X]$ be a uniformly random polynomial of degree at most d. We need to show that for any set $S \subseteq D \times [m]$, the random variables $(\Phi_x(P(x))_i)_{(x,i)\in S}$ are independent.

To see this, first note that for

$$A = \{x \in D \mid \exists i \in [m] \text{ with } (x,i) \in S\},$$

we have that the random variables $(P(x))_{x\in A}$ are uniform and independent. This is because $|A| \leq |S| \leq d+1$, and the evaluations of uniformly random degree d polynomials are uniform and $(d+1)$-wise independent.

Next, we observe that for any x, setting $S_x = \{i \in [m] \mid (x,i) \in S\}$, the random variables $(\Phi(P(x))_i)_{i\in S_x}$ are independent. This is because $P(x)$ is uniformly distributed over $\mathbb{F}_q$, and since Φ is a bijection, the image under Φ of a uniformly random element of $\mathbb{F}_q$ is uniform on $\mathbb{F}_2^m$.

Combining these facts, we get the desired $(d+1)$-wise independence. □

4.2 List-decoding algorithm

We now give a list-decoding algorithm for (general) Raw Reed-Solomon codes, which is interesting in the setting of polynomially small rate and where $|D| = \Omega(q)$.

Lemma 4.3 (Decodability of Raw Reed-Solomon codes)**.** *Let $\mathbb{F}_q, d, m, D, \mathbf{\Phi}$ be as above, and let $C = \mathsf{SimpleRawRS}[\mathbb{F}_q, d, \mathbf{\Phi}]$. Let $\eta > 0$, and suppose*

$$d \leq \eta^2 \cdot \frac{q^{O(\eta^2)}}{q} \cdot |D|.$$

(If $|D| = \Omega(q)$, this is roughly the same as $d \leq \eta^2 \cdot |D|^{O(\eta^2)}$.) Let $n = m \cdot |D|$ be the block-length of C.

Then C is list-decodable from $1/2 - \eta$ fraction errors in time $\mathrm{poly}(n)$ with list size $O(n^2)$.

Proof. We use the natural 2-stage list-decoding strategy for concatenated codes. This will reduce our problem to list-recovery of Reed-Solomon codes, for which we have the following fundamental result.

Theorem 4.4 (List-recovery of Reed-Solomon codes [Sud97, GS99])**.** *Suppose we are given, for each $x \in D$, an "input list" $L_x \subseteq \mathbb{F}_q$ with $|L_x| \leq \ell$. Then we can find, in $\mathrm{poly}(q)$ time, the list of all polynomials $P(X)$ of degree at most d such that:*

$$\Pr_{x\in D}[P(x) \in L_x] \geq \alpha,$$

provided:

$$\alpha \geq \sqrt{\frac{d\ell}{|D|}}.$$

Furthermore, the output list size is at most $O(q^2)$.

Let $w : D \times m \to \mathbb{F}_2$ be a given received word. For $x \in D$, let $w(x) \in \mathbb{F}_2^m$ denote the vector whose ith coordinate is $w(x, i)$.

For each $x \in D$, we define the input-list $L_x \subseteq \mathbb{F}_q$ as follows:

$$L_x = \{u \in \mathbb{F}_q \mid \Delta(\Phi_x(u), w(x)) \leq 1/2 - \eta/2\}.$$

Then for any $c \in C$ with $\Delta(w, c) < 1/2 - \eta$, we have that:

$$\Pr_{x \in D}[c(x) \in L_x] \geq \eta/2.$$

This is because $\mathbf{E}_{x \in D}[\Delta(w(x), c(x))] = \Delta(w, c) < 1/2 - \eta$, and so by Markov's inequality,

$$\Pr_{x \in D}[\Delta(w(x), c(x)) < 1/2 - \eta/2] > \eta/2. \tag{1}$$

We have that each L_x has size

$$\ell = Vol(\text{Ball of radius } (1/2 - \eta/2) \text{ in } \mathbb{F}_2^m) \leq 2^{(1-\Omega(\eta^2))m} = q^{1-\Omega(\eta^2)}.$$

Thus we have that

$$\sqrt{\frac{d\ell}{|D|}} \leq \sqrt{\frac{dq^{1-\Omega(\eta^2)}}{|D|}} \leq \eta/2.$$

Thus the list-recovery algorithm of Theorem 4.4 will find all $P(X) \in \mathbb{F}_q[X]$ of degree at most d such that

$$\Pr_{x \in D}[P(x) \in L_x] \geq \eta/2.$$

By Equation (1), all the codewords we are interested in will be recovered by this procedure.

We summarize the algorithm below:

- Create, for each $x \in D$, an input list $L_x \subseteq \mathbb{F}_q$.
- Use the Reed-Solomon list-recovery algorithm to find all polynomials $P(X)$ for which $P(x) \in L_x$ for a noticeable fraction of $x \in D$.
- For each such polynomial $P(X)$, include the corresponding codeword $c : \mathbb{F}_q \times [m] \to \mathbb{F}_2$ in the output list.

□

4.3 Explicit RawRS codes with good distance

Now we come to the most delicate part: the minimum distance.

In general, a Raw Reed-Solomon code could have minimum distance as small as $1/m = O(1/\log n)$. Indeed VerySimpleRawRS does have small distance. If we take some $\alpha \in \mathbb{F}_q$ for which $\Phi(\alpha)$ has absolute weight equal to 1, then the codeword of a VerySimpleRawRS code which corresponds to the constant polynomial α has minimum distance $O(1/\log n)$.

Nevertheless, the following results shows that OddRawRS and SimpleRawRS have good minimum distance. Our first result shows that OddRawRS has distance $1/2 - o(1)$ for $d = o(q^{1/2})$. Our second result shows that SimpleRawRS has distance about $\frac{1-\epsilon}{2}$ when $d < q^\epsilon$ for any $\epsilon < 1/2$ (and in particular the distance is $1/2 - o(1)$ for $d = q^{o(1)}$).

Lemma 4.5 (Distance of $\mathsf{OddRawRS}$)**.** *Let $\mathbb{F}_q, d, m, \Phi$ be as above, and let $C = \mathsf{OddRawRS}[\mathbb{F}_q, d, \Phi]$.*

Then C has minimum distance at least

$$\left(\frac{1}{2} - \frac{2d}{\sqrt{q}}\right).$$

Proof. The key ingredient in the proof is the Weil bound on additive character sums.

First we recall the field trace function $\mathrm{Tr} : \mathbb{F}_q \to \mathbb{F}_2$. This is an $\mathbb{F}_2$-linear function given by:

$$\mathrm{Tr}(x) = x + x^2 + x^4 + \ldots + x^{2^i} + \ldots + x^{2^{m-1}}.$$

Theorem 4.6 ([Wei48])**.** *Let $\mathrm{Tr} : \mathbb{F}_q \to \mathbb{F}_2$ denote the finite field trace. Let $R(X) \in \mathbb{F}_q[X]$ be a* nonzero *polynomial of degree at most d with only* odd degree *monomials. Then:*

$$\left|\sum_{x \in \mathbb{F}_q} (-1)^{\mathrm{Tr}(R(x))}\right| \leq (d-1)\sqrt{q}.$$

It says that for low degree polynomials R with only odd degree monomials, $\mathrm{Tr}(R(x))$ is approximately uniformly distributed over $\mathbb{F}_2$. The hypothesis about odd degree is needed to avoid pathological situations where $\mathrm{Tr}(R(x))$ is constant (for example, $\mathrm{Tr}(x + x^2) = 0$ for all x). The statement above can be found in [Sch06, Chapter II.2, Theorem 2E]. Elementary proofs were given by Stepanov, Schmidt and Bombieri (see [Sch06, Mor93, Kop10] for expositions).

We also need some simple facts about Tr.

- Every $\mathbb{F}_2$-linear function $g : \mathbb{F}_q \to \mathbb{F}_2$ is of the form $g(x) = \mathrm{Tr}(\beta x)$ for some $\beta \in \mathbb{F}_q$.
- $\mathrm{Tr}(y) = \mathrm{Tr}(y^2)$ for all $y \in \mathbb{F}_q$.

By the first fact above, there are $\beta_1, \ldots, \beta_m \in \mathbb{F}_q$ such that $\Phi : \mathbb{F}_q \to \mathbb{F}_2^m$ is given by:

$$\Phi(y) = (\mathrm{Tr}(\beta_1 y), \mathrm{Tr}(\beta_2 y), \ldots, \mathrm{Tr}(\beta_m y)).$$

Since Φ is injective, we get that $\beta_1, \ldots, \beta_m$ are linearly independent over $\mathbb{F}_2$, and thus are a basis for $\mathbb{F}_q$ over $\mathbb{F}_2$.

Let $c : \mathbb{F}_q \times [m] \to \mathbb{F}_2$ be a nonzero codeword. We break it into m functions $c_1, c_2, \ldots, c_m : \mathbb{F}_q \to \mathbb{F}_2$ given by:

$$c_i(x) = c(x, i).$$

It will turn out that each c_i is a nonzero codeword of a dual-BCH code.

Let $P(X)$ be the polynomial underlying c. We have $\deg(P(X)) \leq d$. Let $P(X) = \sum_{j=0}^{d} \gamma_j X^j$.

By definition of $\mathsf{OddRawRS}$, we have $\Phi_x(y)_i = \mathrm{Tr}(\beta_i x y^2) = \mathrm{Tr}(\beta_i x P(x)^2)$. The crucial point is that

$$\begin{aligned}
c_i(x) = \Phi_x(P(x))_i &= \mathrm{Tr}(\beta_i x P(x)^2) \\
&= \mathrm{Tr}\left(\beta_i x \left(\sum_{j \leq d} \gamma_j^2 x^{2j}\right)\right) \\
&= \mathrm{Tr}\left(\beta_i \sum_{\ell \leq 2d+1, \ell \text{ odd}} \gamma_{(\ell-1)/2}^2 x^{\ell}\right) \\
&= \mathrm{Tr}(R_i(x)),
\end{aligned}$$

where:

$$R_i(X) = \sum_{\ell \leq 2d+1, \ell \text{ odd}} \beta_i \gamma_{(\ell-1)/2}^2 X^\ell$$

is a *nonzero* polynomial of degree at most $2d+1$ with only odd degree monomials. This allows us to apply the Weil bound (Theorem 4.6) directly. It tells us that for *all* $i \in [m]$,

$$\left| \Pr_x[\mathrm{Tr}(R_i(x)) = 0] - \Pr_x[\mathrm{Tr}(R_i(x)) = 1] \right| \leq \frac{2d}{\sqrt{q}} + \frac{1}{q}.$$

So, using wt to denote the relative weight,

$$\mathsf{wt}(c_i) = \Pr_{x \in D}[\mathrm{Tr}(R_i(x)) \neq 0] \geq \left(1/2 - \frac{d}{\sqrt{q}} - \frac{1}{2q}\right).$$

Averaging over all i, we get that

$$\mathsf{wt}(c) = \mathbf{E}_{i \in [m]}[\mathsf{wt}(c_i)] \geq \left(1/2 - \frac{d}{\sqrt{q}} - \frac{1}{2q}\right).$$

Thus the minimum distance of C is at least that quantity, as desired. □

Lemma 4.7 (Distance of $\mathsf{SimpleRawRS}$)**.** *Let $\mathbb{F}_q, d, m, \Phi$ be as above, and let $C = \mathsf{SimpleRawRS}[\mathbb{F}_q, d, \Phi]$. Then C has minimum distance at least*

$$\left(1 - \frac{\log(d+1)}{\log q}\right)\left(\frac{1}{2} - \frac{d}{\sqrt{q}}\right).$$

Proof. The proof is very similar to the previous one.

Again we have a basis $\beta_1, \ldots, \beta_m$ of $\mathbb{F}_q$ over $\mathbb{F}_2$ such that

$$\Phi(y) = (\mathrm{Tr}(\beta_1 y), \mathrm{Tr}(\beta_2 y), \ldots, \mathrm{Tr}(\beta_m y)).$$

Let c be a nonzero codeword. We define $c_1, \ldots, c_m : D \to \mathbb{F}_2$ as before:

$$c_i(x) = c(x, i).$$

Here again we will get that the c_i are codewords of the dual-BCH code. However, unlike the previous proof, here it might be the case that some c_i is identically 0. We will show that at most $\log_2(d+1)$ of these c_i are identically 0, and the remaining c_i have weight at least $\left(\frac{1}{2} - \frac{d}{\sqrt{q}}\right)$. This will imply that:

$$\mathsf{wt}(c) \geq \mathbf{E}_{i \in [m]}[\mathsf{wt}(c_i)] \geq \left(1 - \frac{\log(d+1)}{\log q}\right)\left(\frac{1}{2} - \frac{d}{\sqrt{q}}\right),$$

and thus the minimum distance of C is at least that quantity, completing the proof.

Let $P(X)$ be the polynomial underlying c. We have $\deg(P(X)) \leq d$. Let $P(X) = \sum_{j=0}^{d} \gamma_j X^j$. By construction, $c_i(x) = \mathrm{Tr}(\beta_i x P(x))$. The polynomial $\beta_i X P(X)$ may have monomials of even degree, and so we cannot directly apply Theorem 4.6 to it. Instead we will reduce the even degree monomials using the identity $\mathrm{Tr}(y^2) = \mathrm{Tr}(y)$, and then hope that the reduction does not leave us with the zero polynomial.

$$\begin{aligned}
\beta_i X P(X) &= \beta_i \left(\sum_{j=1}^{d+1} \gamma_{j-1} X^j \right) \\
&= \sum_{j=1}^{d+1} \beta_i \gamma_{j-1} X^j \\
&= \sum_{\ell \le d+1, \ell \text{ odd}} \left(\sum_{r \ge 0, 2^r \ell \le d+1} \beta_i \gamma_{(\ell \cdot 2^r - 1)} X^{\ell \cdot 2^r} \right),
\end{aligned}$$

where in the last equality, we grouped all the powers of X according to the largest odd factor of the exponent (for example, $X^3, X^6, X^{12}, X^{24}, \ldots$ are all in the same group).

Thus for every $x \in \mathbb{F}_q$, we have:

$$\begin{aligned}
\mathrm{Tr}(\beta_i x P(x)) &= \sum_{\ell \le d+1, \ell \text{ odd}} \left(\sum_{r \ge 0, 2^r \ell \le d+1} \mathrm{Tr}(\beta_i \gamma_{(\ell \cdot 2^r - 1)} x^{\ell \cdot 2^r}) \right) \\
&= \sum_{\ell \le d+1, \ell \text{ odd}} \left(\sum_{r \ge 0, 2^r \ell \le d+1} \mathrm{Tr}\left(\left(\beta_i^{1/2^r} \gamma_{(\ell \cdot 2^r - 1)}^{1/2^r} x^{\ell} \right)^{2^r} \right) \right) \\
&= \sum_{\ell \le d+1, \ell \text{ odd}} \left(\sum_{r \ge 0, 2^r \ell \le d+1} \mathrm{Tr}\left(\beta_i^{1/2^r} \gamma_{(\ell \cdot 2^r - 1)}^{1/2^r} x^{\ell} \right) \right) \qquad \text{Since } \mathrm{Tr}(y^{2^r}) = \mathrm{Tr}(y) \\
&= \mathrm{Tr}\left(\sum_{\ell \le d+1, \ell \text{ odd}} \left(\sum_{r \ge 0, 2^r \ell \le d+1} \beta_i^{1/2^r} \gamma_{(\ell \cdot 2^r - 1)}^{1/2^r} \right) x^{\ell} \right) \\
&= \mathrm{Tr}\left(\sum_{\ell \le d+1, \ell \text{ odd}} E_\ell(\beta_i) x^{\ell} \right),
\end{aligned}$$

where $E_\ell : \mathbb{F}_q \to \mathbb{F}_q$ is the function

$$E_\ell(y) = \sum_{r \ge 0, 2^r \ell \le d+1.} y^{1/2^r} \gamma_{(\ell \cdot 2^r - 1)}^{1/2^r}.$$

Let $R_i(X) \in \mathbb{F}_q[X]$ be given by:

$$R_i(X) = \sum_{\ell \le d+1, \ell \text{ odd}} E_\ell(\beta_i) X^{\ell}.$$

Summarizing, we have

$$c_i(x) = \mathrm{Tr}(R_i(x)).$$

Since $P(X)$ is a nonzero polynomial, some coefficient $\gamma_{j_0} \neq 0$. Let $j_0 = \ell_0 \cdot 2^{r_0} - 1$, where ℓ_0 is odd. Observe that E_{ℓ_0} satisfies:

1. E_{ℓ_0} is $\mathbb{F}_2$-linear,
2. $E_{\ell_0}(y) = A(y^{1/2^\rho})$ for some nonnegative integer ρ and some polynomial $A(Z) \in \mathbb{F}_q[Z]$ of degree at most $d+1$,
3. $A(Z)$ has some power of γ_{j_0} as a coefficient of some monomial, and is thus a nonzero polynomial.

These three facts imply that E_{ℓ_0} can vanish on at most $\log_2(d+1)$ of the β_i. Indeed, since the β_i are linearly independent, if E_{ℓ_0} vanishes on t of them, then by linearity we get that E_{ℓ_0} vanishes on their span, which has 2^t points. However E_{ℓ_0} cannot have more than $d+1$ roots (since A has degree at most $d+1$). Thus $t \leq \log_2(d+1)$.

In particular, this means that at most $\log_2(d+1)$ of the $R_i(X)$ are identically zero.

Fix an i where $R_i(X)$ is not identically 0. We have that $R_i(X)$ is a nonzero polynomial of degree at most $d+1$ with only monomials of odd degree. Thus Theorem 4.6 applies. It tells us that

$$\left| \Pr_{x\in D}[\mathrm{Tr}(R_i(x)) = 0] - \Pr_{x\in D}[\mathrm{Tr}(R_i(x)) = 1] \right| \leq \frac{d+1}{\sqrt{q}} + \frac{1}{q}.$$

Then we get:

$$\mathsf{wt}(c_i) = \Pr_{x\in D}[\mathrm{Tr}(R_i(x)) \neq 0] \geq \left(\frac{1}{2} - \frac{d+1}{2\sqrt{q}} - \frac{1}{q}\right).$$

Since there are at least $\log_2 q - \log_2(d+1)$ such i, we get the desired claim about the weight of c. This completes the proof of the minimum distance of SimpleRawRS codes. □

4.4 Discussion

1. Consider (general) Raw Reed-Solomon codes with $D = \mathbb{F}_q$ and $d \leq q^{0.01}$. Somewhat surprisingly, even though these codes need not have $\Omega(1)$ distance, they all have polynomial list-size for list-decoding upto radius almost $1/2$.

   Indeed, since the linear bijections Φ_x are completely arbitrary, we can choose them so that some particular polynomial $P(X)$ has the property that $\Phi_x(P(x))$ has Hamming weight ≤ 1 for all $x \in D$. The codeword of C corresponding to $P(X)$ will have relative Hamming weight $1/m = \Theta(1/\log n)$. However, as the list-decodability implies, the underlying algebraic structure somehow forces that one cannot choose the $(\Phi_x)_{x\in\mathbb{F}_q}$ so that this happens for many other $P(X)$.

2. Thommesen [Tho83] showed that if we choose the entries of $\mathbf{\Phi} = (\Phi_x)_{x\in\mathbb{F}_q}$ independently and uniformly at random (i.e., each Φ_x is an independently chosen uniformly random $\mathbb{F}_2$-linear bijection from $\mathbb{F}_q$ to $\mathbb{F}_2^m$), then for all d the resulting Raw Reed-Solomon code $C = \mathsf{RawRS}(\mathbb{F}_q, d, D, \mathbf{\Phi})$, for arbitrary D, meets the Gilbert-Varshamov bound[16] with high probability. In particular, even for $d = \Omega(|D|)$ (when the rate is $\Omega(1)$), there are Raw Reed-Solomon codes that have distance $\Omega(1)$. It is easy to see that no Simple Raw Reed-Solomon code has this property.

   Finding an explicit such code seems like a deep and very interesting open question.

3. VerySimpleRawRS and SimpleRawRS are closely related, yet have very different minimum distances. The results about SimpleRawRS explain the structure of VerySimpleRawRS. VerySimpleRawRS is

[16] The Gilbert-Varshamov bound $R = 1 - H(\delta)$ is the best *known* rate for codes with relative distance δ. This result is not constructive - deterministically constructing codes that meet this bound is a central open question.

the space spanned by SimpleRawRS along with codewords corresponding to the constant polynomials. SimpleRawRS has good minimum distance, it is only the small dimensional space of constant polynomials that spoils the minimum distance. This also explains why VerySimpleRawRS has good list-decodability despite having bad distance.

4. An important fact underlying our analysis of the distance of SimpleRawRS and OddRawRS is that Raw Reed-Solomon codes are just a bunch of (correlated) dual-BCH codewords written together. We don't know how to decode dual-BCH codes efficiently, but if we take about $\log n$ dual-BCH codewords together, then this resulting code magically can be decoded, while still retaining the good distance and dual distance of dual-BCH codes.

4.5 Proof of Theorem 4.1

We can now put everything together and prove Theorem 4.1.

Proof. Take $q = 2^m$, and let $d = q^\alpha$ be an even number. Let $\Phi : \mathbb{F}_q \to \mathbb{F}_2^m$ be an arbitrary $\mathbb{F}_2$-linear bijection. Take $C = \mathsf{OddRawRS}[\mathbb{F}_q, d, \Phi]$. Recall that this code is the Raw Reed-Solomon code with evaluation domain $D = \mathbb{F}_q \setminus \{0\}$ with certain special $\Phi_x : \mathbb{F}_q \to \mathbb{F}_2^m$.

We now specify a linear encoding map for C. We take the encoding map $\mathrm{Enc} : \mathbb{F}_2^k \to \mathbb{F}_2^n$ to be the one which partitions the k input bits into blocks of size m, interprets the ith block as specifying (in an $\mathbb{F}_2$-linear way) the coefficient of the X^i monomial in a polynomial $P(X)$, and outputs the codeword c of C corresponding to the polynomial $P(X)$. Then clearly $\mathrm{Enc}^{\mathrm{trunc}}$ is simply the encoding map of $C' = \mathsf{OddRawRS}[\mathbb{F}_q, d/2, \Phi]$.

Since OddRawRS is an instance of RawRS, we can apply Lemma 4.2. It gives us the following basic properties of C:

1. The blocklength n of C equals $m \cdot (2^m - 1)$.
2. The dimension k of C equals $m \cdot (d+1) \geq m \cdot q^\alpha \geq m^{1-\alpha} \cdot n^\alpha$.
3. The absolute dual distance of C is at least $d + 2 \geq q^\alpha \geq \left(\frac{n}{\log n}\right)^\alpha$.

Apply the same lemma to C' tells us that the absolute dual distance of C' is at least $d/2+2 \geq \Omega\left(\left(\frac{n}{\log n}\right)^\alpha\right)$.

Next we invoke Lemma 4.5. This is the place where we use the specific structure of *Odd* Raw Reed-Solomon codes. We get that C has distance at least:

$$\delta = \frac{1}{2} - O\left(\frac{q^\alpha}{q}\right) \geq \frac{1}{2} - n^{-\Omega(1)}.$$

Next we invoke Lemma 4.3. Set $\eta = b\sqrt{\alpha}$ for some absolute constant b. Since $|D| = q - 1$, we get that

$$d = q^\alpha \leq \eta^2 q^{O(\eta^2)} \leq \eta^2 q^{O(\eta^2)} \cdot \frac{|D|}{q}.$$

Thus C can be list-decoded from $1/2 - \eta$ fraction errors in time $\mathrm{poly}(q) \leq \mathrm{poly}(n)$. As an immediate consequence, since $1/2 - \eta > \delta/2$, we get that C can be unique decoded from $\delta/2 > 1/4 - o(1) > p$ fraction errors in polynomial time: we simply run the list-decoder and find the unique (if any) element of the output list which is within distance p from the received word.

The list-size guaranteed by Lemma 4.3 only implies that the list-size is at most $\mathrm{poly}(n)$, which is weaker than what we want. However now we only seek a combinatorial bound on the list-size, and this follows immediately from the Johnson bound [Joh62], which states that binary codes with minimum distance $\geq 1/2 - o(1)$ have list-size at most $O(1/\epsilon^2)$ for list-decoding from $(1/2-\epsilon)$-fraction errors.

This completes the proof of the theorem. □

5 Stochastic control codes

In this section we consider a more stringent notion of stochastic codes that are decodable from t errors. We will also require that such codes have an additional "pseudorandom property", namely that for every message $m \in \{0,1\}^k$, $\mathrm{Enc}(m, U_d)$ is pseudorandom for small space ROBPs.

Definition 5.1 (Pseudorandom stochastic Codes decodable from errors)**.** *Let k, n, d be parameters and let $\mathrm{Enc} : \{0,1\}^k \times \{0,1\}^d \to \{0,1\}^n$ be a function. We say that Enc is an encoding function for a stochastic code that is:*

- *ϵ-**pseudorandom** for a class $\mathcal{C}$ of functions from n bits to one bit, if for every $m \in \{0,1\}^k$, $\mathrm{Enc}(m, U_d)$ is ϵ-pseudorandom for $\mathcal{C}$.*
- ***decodable from** t **errors**, if $t \in [n]$, and there exists a function $\mathrm{Dec} : \{0,1\}^n \to \{0,1\}^k$ such that for every $m \in \{0,1\}^k$, $s \in \{0,1\}^d$, and $e \in \{0,1\}^n$ with Hamming weight at most t, $\mathrm{Dec}(\mathrm{Enc}(m,s) \oplus e) = m$.*
- *L-**list-decodable from** t**-errors**, if the function Dec is allowed to output a list of size at most L, and for every $m \in \{0,1\}^k$, $s \in \{0,1\}^d$, and $e \in \{0,1\}^n$ with Hamming weight at most t, $\mathrm{Dec}(\mathrm{Enc}(m,s) \oplus e) \ni m$.*

*A code has **encoding time** [resp. **decoding time**] $T(\cdot)$, if Enc [resp. Dec] can be computed in time $T(k+n+d)$. (Naturally, this makes sense only for a family of encoding and decoding functions with varying block length n, message length $k(n)$ and seed length $d(n)$).*

Remark 5.2 (This notion is only interesting for pseudorandom codes)**.** *We remark that the notion of stochastic codes decodable (or list-decodable) from t-errors is not interesting by itself. This is because for any such code $\mathrm{Enc}(m,s)$, we can define a standard (not stochastic) code $\mathrm{Enc}'(m) = \mathrm{Enc}(m, s')$ for some fixed seed s', and this code will be decodable (or list decodable) from t-errors.*

This means that designing stochastic codes that are decodable from t errors is a harder task *than designing standard codes that are decodable from errors, and we don't gain (and in fact make our task more difficult) by allowing Enc to receive a seed.*

This notion of codes decodable from errors becomes interesting when it is coupled with the pseudorandomness requirement. Loosely speaking, one can think of such codes as "standard codes" with an additional pseudorandomness property.

Remark 5.3 (The use of this notion in past work)**.** *Similar notions appear in [GS16, SS16]. Specifically, Guruswami and Smith [GS16] considered a notion similar to "list-decodable from errors" with the stronger requirement that the decoding function needs to produce the randomness s, in addition to the message m.*

Shaltiel and Silbak [SS16] referred to this stronger requirement as "strongly list-decodable" and to the weaker notion defined here as "weakly list-decodable". The fact that the weaker notion (in which decoding does not need to produce the randomness) suffices for the intended application of stochastic codes for bounded channels, was key in [SS16] (as this weaker codes are easier to construct). The same also holds for this paper, as the list-decoding algorithms that we construct will not be able to reproduce the randomness s used by the encoding.

The main result of this section (that is stated in the theorem below) is a construction of stochastic codes that are pseudorandom for small space ROBPs. We plan to use these codes to encode very short strings, and so, their rate is not that important to us. The construction uses the OddRawRS of Section 4.

The theorem below gives a construction of a stochastic code that will be used as a "control code" in the construction of Section 6. We will use this "control code" in the proof of Theorem 1.4, which is our main construction for bounded channels.

Theorem 5.4 (Stochastic control codes for space $n^{\Omega(1)}$, with list-decoding up to $\frac{1}{2}$)**.** *For every constant $\beta > 0$ there exists a constant $0 < \alpha \le 0.1$ such that for every sufficiently large m, setting $n = (2^m - 1) \cdot m$, $k = n^\alpha$, $d = n \log n$, and $s = \frac{n^\alpha}{\log^3 n}$, there is a stochastic code $\mathrm{Enc} : \{0,1\}^k \times \{0,1\}^d \to \{0,1\}^n$ that is:*

- 2^{-s}*-pseudorandom for any-order space s ROBPs.*
- *For every constant $p < 1/4$,* Enc *is decodable from pn errors in time* $\mathrm{poly}(n)$.
- *For every constant $\epsilon > \beta$,* Enc *is $O(1/\epsilon^2)$-list decodable from $(\frac{1}{2} - \epsilon) \cdot n$ errors in time* $\mathrm{poly}(n)$.
- *There exists a constant c, such that* Enc *can be computed in time n^c. Furthermore, encoding n^c inputs takes "amortized time" $O(n \cdot \log^2 n)$, namely, for every $(m_1, s_1), \dots, (m_{n^c}, s_{n^c}) \in \{0,1\}^k \times \{0,1\}^d$, computing $(\mathrm{Enc}(m_i, S_i))_{(i \in [n^c])}$ takes time $n^c \cdot O(n \cdot \log^2 n)$.*

In the construction proving Theorem 1.4 we will apply the stochastic control code many times, and this is why we care about amortized encoding time (that can be made quasilinear) rather than the time of encoding one message.

We can also get a different tradeoff that gives pseudorandomness for larger space. However, this comes with a cost of decoding only from $p_0 \cdot n$ errors for some small constant $p_0 > 0$ (rather than a number of errors that approaches $\frac{1}{2} \cdot n$). The encoding algorithm for this code is also less efficient than the one in Theorem 5.4, and we don't get encoding in amortized linear time. The theorem below will be used in the proof of Theorem 1.5.

Theorem 5.5 (Stochastic control codes for space $n/\mathrm{polylog}(n)$, that decode from few errors)**.** *There exist constants $p_{\max} > 0$, and $R > 0$ such that for every sufficiently large m, setting $n = 8^m - 8$, $k = Rn$, $d = n \log n$, and $s = n/\log^2 n$ there is a stochastic code $\mathrm{Enc} : \{0,1\}^k \times \{0,1\}^d \to \{0,1\}^n$ that is:*

- 2^{-s}*-pseudorandom for any-order space s ROBPs.*
- Enc *is decodable from $p_{\max} \cdot n$ errors in time* $\mathrm{poly}(n)$.
- Enc *is encodable in time* $\mathrm{poly}(n)$.

In the remainder of the section we prove Theorem 5.4 and Theorem 5.5. In Section 5.1 we revisit the idea of "bounded independence plus noise" of [HLV18, LV17, FK18]. We state and prove a quantitative variant of this approach that will be used in our proof. In Section 5.2 we use the method of "bounded independence plus noise" to transform linear codes with certain additional properties into stochastic control codes that are pseudorandom for ROBPs. Finally, in Section 5.3 we prove Theorem 5.4 (by relying on the OddRawRS of Section 4) and Theorem 5.5 (by relying on algebraic geometric codes of Garcia and Stichtenoth [GS96]).

5.1 Bounded independence plus low weight noise fools ROBPs

Haramaty, Lee and Viola [HLV18] showed that "k-wise independence plus low weight noise" is pseudorandom for certain classes of distinguishers. Specifically, they consider xoring a k-wise independent distribution D_k^n on n bits, with low weight noise, chosen according to BSC_η^n for some small $\eta > 0$. They show that this distribution, namely $D_k^n \oplus \mathrm{BSC}_\eta^n$ (where the two distributions are independent) is pseudorandom for small space ROBPs if k is sufficiently larger than $n^{2/3}$, and $\eta > 0$ is a positive constant (that can be arbitrarily small).

This result is specifically appealing as this gives a distribution that is pseudorandom for ROBPs, regardless of the order in which they read the n bits. However, it requires a very large seed length, of at least $n^{2/3}$, (even for generating the distribution D_k^n).

We are interested in constructing stochastic control codes (which combine requirements from coding theory and pseudorandomness) and will make use of the particular structure of the distribution of [HLV18] (in addition to their pseudorandomness properties). More specifically, the fact that w.h.p. BSC_η has low hamming weight, will be important in our intended application.

Unfortunately, for our intended application, taking $k > n^{2/3}$ is too large, and we need k to be smaller. Subsequent work [LV17, FK18], gives pseudorandom distribution in which k is much smaller, but they use a different distribution in which the "noise distribution" BSC_η is replaced by distributions which have large hamming weight of $\approx \frac{n}{4}$). Such large weight is not useful for our application.[17]

In this paper we observe that the ideas and technique use by Forbes and Kelly [FK18] to reduce the amount of independence for "high weight noise", can also be applied in the case of noise with low hamming weight. This is stated precisely in the following theorem.

Theorem 5.6 (Improved analysis for [HLV18])**.** *For every integers n, s and $\epsilon, \eta > 0$, the distribution $D_k^n \oplus \mathrm{BSC}_\eta^n$ where D_k^n is independent of BSC_η^n, and is k-wise independent over $\{0,1\}^n$, for $k = O(\frac{s+\log n+\log(1/\epsilon)}{\eta})$, is ϵ-pseudorandom for any-order space s ROBPs.*

Note that Theorem 5.6 kicks in, whenever $k > \log n$ whereas the previous analysis by [HLV18] only kicked in if $k \geq n^{2/3}$.

Proof. Forbes and Kelly [FK18, Lemma 6.3] show that the distribution $D_{2k}^n \oplus (T_k^n \wedge U_n)$ where the three distributions are independent, and:

- D_{2k}^n is $2k$-wise independent over $\{0,1\}^n$.
- T_k^n is k-wise independent over $\{0,1\}^n$.

is ϵ-pseudorandom for width $w = 2^s$ ROBPs, with $\epsilon = \frac{nw}{2^{k/2}}$. We first note that $\mathrm{BSC}_\eta^n = \mathrm{BSC}_{2\eta}^n \wedge U_n$. Thus, if we change D_k^n to D_{2k}^n (which we can do because of the $O(\cdot)$ notation in our statement) we can think of our target distribution $D_{2k}^n \oplus \mathrm{BSC}_\eta^n$ as $D_{2k}^n \oplus (\mathrm{BSC}_{2\cdot\eta}^n \wedge U_n)$. Consequently, in order to prove our result, we need to show that the analysis of [FK18] can be carried out in case T_k^n is replaced by $\mathrm{BSC}_{2\eta}$. This is indeed the case, and the analysis gives $\epsilon = nw \cdot (1-2\eta)^{k/2}$ under this modification. This is sufficient to derive our result.

On an intuitive level, this follows because the distribution T_k^n is only used to argue that if $\alpha \in \{0,1\}^n$ is a "Fourier coefficient" with weight k, then $\Pr[T_k^n \wedge \alpha = 0] = (\frac{1}{2})^k$. The distribution $\mathrm{BSC}_{2\eta}^n$ (which plays the role of T_k^n in our case) gives the similar (though slightly weaker) bound of $\Pr[\mathrm{BSC}_{2\eta}^n \wedge \alpha = 0] = (1-2\eta)^k$, and this suffices for the argument.

More precisely, inspecting the proof of [FK18, Lemma 6.3], one can observe that the only place where a specific property of the distribution T_k^n is used is at the final equality in the proof of Lemma 6.2, and that

[17] Lee and Viola [LV17] and Forbes and Kelly [FK18] consider the noise distribution $T_k^n \wedge U_n$ (where T_k^n is a k-wise independent distribution). Their motivation is that using this approach, one can think of T_k^n as selecting approximately $n/2$ of the n indices, and then placing uniform bits on these $n/2$ indices. This view enables a recursive construction in which the $n/2$ uniform bits are replaced with pseudorandom bits, and this approach can yield pseudorandom generators with polylogarithmic seed [FK18].

replacing T_k^n with $\mathrm{BSC}_{2\eta}^n$ yields a version of Lemma 6.2. in which the term $(\frac{1}{2})^k$. is replaced by $(1-2\eta)^k$. Finally, Lemma 6.2 is used to derive the last inequality in Lemma 6.3. and substituting the modified quantity gives the final result. (The rest of the proof goes through unchanged). $\square$

5.2 Linear codes with large dual distance yield pseudorandom stochastic codes

In this section we show that linear codes with large dual distance can be used to construct stochastic control codes.

In the definition below we define a function $\mathrm{BSC}_\eta^n(\cdot)$ which when given uniform input, generates the distribution BSC_η^n. We then consider a truncated version $\mathrm{BSC}_\eta^{n,\mathrm{trunc}}(\cdot)$ which evaluates to 0^n if the generated string has hamming weight larger than $2\eta \cdot n$. This is done to guarantee that with probability one, over a uniform input, $\mathrm{BSC}_\eta^{n,\mathrm{trunc}}$ produces a string with hamming weight $\leq 2\eta \cdot n$.

Definition 5.7 (Generating and truncating BSC)**.** *For an integer k, $\eta = \frac{1}{2^k}$, and an integer n, we define the function $\mathrm{BSC}_\eta^n : \{0,1\}^{n\cdot\log(1/\eta)} \to \{0,1\}^n$ by $\mathrm{BSC}(s)_i = 1$ iff $s_{(i-1)\cdot\log(1/\eta)+1}, \ldots, s_{i\cdot\log(1/\eta)} = 1$, so that $\mathrm{BSC}_\eta^n(U_{n\cdot\log(1/\eta)})$ is the distribution BSC_η^n.*

The truncated version $\mathrm{BSC}_\eta^{n,\mathrm{trunc}} : \{0,1\}^{n\cdot\log(1/\eta)} \to \{0,1\}^n$ is defined as follows: Given $s \in \{0,1\}^{n\cdot\log(1/\eta)}$, if $\mathrm{BSC}_\eta^n(s)$ has hamming weight larger than $2\eta \cdot n$, then $\mathrm{BSC}_\eta^{n,\mathrm{trunc}}(s)$ is set to 0^n, and otherwise, it is set to $\mathrm{BSC}_\eta^n(s)$.

Note that by a multiplicative Chernoff bound, the statistical distance between $\mathrm{BSC}_\eta^n(U_{n\cdot\log(1/\eta)})$ and $\mathrm{BSC}_\eta^{n,\mathrm{trunc}}(U_{n\cdot\log(1/\eta)})$ is $2^{-\Omega(n)}$. (This will allow us to replace the former by the latter).

The next definition shows how to convert a linear code Enc into a stochastic code Enc_η (and we soon show that Enc_η is pseudorandom for any-order small space ROBPs).

Definition 5.8 (stochastic control codes from linear codes)**.** *Given a function $\mathrm{Enc} : \{0,1\}^k \to \{0,1\}^n$, we define $\mathrm{Enc}^{\mathrm{trunc}} : \{0,1\}^{k/2} \to \{0,1\}^n$ by $\mathrm{Enc}^{\mathrm{trunc}}(x) = \mathrm{Enc}(x \circ 0^{k/2})$.*

Given a function $\mathrm{Enc} : \{0,1\}^k \to \{0,1\}^n$ and $\eta > 0$, we define $d = k/2 + n \cdot \log(1/\eta)$, and the function $\mathrm{Enc}_\eta : \{0,1\}^{k/2} \times \{0,1\}^d \to \{0,1\}^n$ as follows: Given inputs $m \in \{0,1\}^{k/2}$ and $s \in \{0,1\}^d$, we interpret s as a pair $s = (s_1, s_2)$ where $s_1 \in \{0,1\}^{k/2}$ and $s_2 \in \{0,1\}^{n\cdot\log(1/\eta)}$, and define:

$$\mathrm{Enc}_\eta(m, s) = \mathrm{Enc}(s_1 \circ m) \oplus \mathrm{BSC}_\eta^{n,\mathrm{trunc}}(s_2)$$

The following lemma shows that if Enc is a linear encoding map for a code with large dual distance, then Enc_η is a stochastic code which is pseudorandom, and inherits the decoding capabilities of Enc.

We plan to apply the stochastic code on many inputs, and are therefore interested in the *amortized* encoding time. The last item of the following lemma says that if k is sufficiently small compared to n, encoding Enc_η takes amortized *quasilinear time* even if one evaluation of Enc takes polynomial (but not necessarily quasilinear) time.

Lemma 5.9. *Let $\mathrm{Enc} : \mathbb{F}_2^k \to \mathbb{F}_2^n$ be a linear function. Let $\eta > 0$, and $d = k/2 + n \cdot \log(1/\eta)$.*

- *If the linear code $C' = \mathrm{Enc}^{\mathrm{trunc}}(\mathbb{F}_2^{k/2})$ has dual distance $r \geq \frac{10\log n}{\eta}$, then $\mathrm{Enc}_\eta : \{0,1\}^{k/2} \times \{0,1\}^d$ is 2^{-s}-pseudorandom for any-order, space s ROBPs, for $s = \Omega(r \cdot \eta)$.*
- *If Enc is decodable [resp. L-list decodable] from pn errors, then Enc_η is decodable [resp. L-list decodable] from $(p - 2\eta) \cdot n$ errors. Furthermore, the decoding time for Enc_η is the same as that of Enc.*

- *Computing* Enc_η *takes time* $O(n \cdot \log(1/\eta))$ *plus the time it takes to compute* Enc.[18]
- *If* $k \le n^{0.1}$ *and* Enc *can be computed in time* n^c *for some constant c, then computing* Enc_η *on* n^c *pairs* $(m_1, s_1), \dots, (m_{n^c}, s_{n^c}) \in \{0,1\}^{k/2} \times \{0,1\}^d$*, takes time* $n^c \cdot O(n \cdot (\log^2 n + \log(1/\eta)))$ *(that is* amortized time $O(n \cdot (\log^2 n + \log(1/\eta)))$*).*

Proof. We start with the first item. By Lemma 3.15 we have that $\mathrm{Enc}^{trunc}(U_{k/2})$ is $(r-1)$-wise independent. Let $\mathrm{Enc}^{\mathrm{suf}} : \mathbb{F}_2^{k/2} \to \mathbb{F}_2^n$ be the function $\mathrm{Enc}^{\mathrm{suf}}(x) = \mathrm{Enc}(0^{k/2} \circ x)$. We first observe that using the linearity of Enc:

$$\mathrm{Enc}(s_1 \circ m) = \mathrm{Enc}^{\mathrm{trunc}}(s_1) \oplus \mathrm{Enc}^{\mathrm{suf}}(m)$$

It follows that for every $m \in \{0,1\}^{k/2}$, $\mathrm{Enc}(U_{k/2} \circ m) = \mathrm{Enc}^{\mathrm{trunc}}(U_{k/2}) \oplus \mathrm{Enc}^{\mathrm{suf}}(m)$ is also $(r-1)$-wise independent. Therefore,

$$\mathrm{Enc}_\eta(m, U_d) = \mathrm{Enc}(U_{k/2} \circ m) \oplus \mathrm{BSC}_\eta^{n,\mathrm{trunc}}(U_{n \cdot \log(1/\eta)}),$$

is $2^{-\Omega(n)}$-close to a distribution of the form $D_{r-1}^n \oplus \mathrm{BSC}_\eta^n$, for some $(r-1)$-wise independent distribution D_{r-1}^n. Therefore, $\mathrm{Enc}_\eta(m, U_d)$ is a distribution that is very close to "$(r-1)$-wise independent plus low weight noise". By Theorem 5.6 with $\epsilon = 2^{-2s}$, $D_{r-1}^n \oplus \mathrm{BSC}_\eta^n$ is 2^{-2s}-pseudorandom for any-order ROBPs with space $\Omega(r \cdot \eta) - \log n - 2s \ge s$ for our choice of parameters.

The distribution $\mathrm{Enc}_\eta(m, U_d)$ is therefore ϵ'-pseudorrandom for any-order ROBPs with space s, for $\epsilon' = 2^{-2s} + 2^{-\Omega(n)} \le 2^{-s}$ by noting that $s \le r \le n$ and choosing the constant hidden in the definition of s to be sufficiently small.

For the second item, note that if we encode a message m, by $\mathrm{Enc}_\eta(m, s) = \mathrm{Enc}(s_1 \circ m) \oplus \mathrm{BSC}_\eta^{n,\mathrm{trunc}}(s_2)$ and xor it with an error vector $e \in \{0,1\}^n$ is of hamming weight $(p - 2\eta) \cdot n$, then (as the hamming weight of $\mathrm{BSC}_\eta^{n,\mathrm{trunc}}(s_2)$ is at most $2\eta \cdot n$) we obtain a string that is within hamming distance $(p-2\eta) \cdot n + 2\eta \cdot n = pn$ from $\mathrm{Enc}(s_1 \circ m)$. Consequently, decoding (or list decoding) is guaranteed to decode (or list decode) the message $s_1 \circ m$ from which we can recover m.

The third item follows directly by construction.

For the fourth item, we note that by the previous item, we only need to show how given $x_1, \dots, x_{n^c} \in \{0,1\}^k$, we can compute $\mathrm{Enc}(x_1), \dots, \mathrm{Enc}(x_{n^c})$ in time $n^c \cdot O(n \cdot \log^2 n)$. Using the fact that Enc can be computed in time n^c, gives that in time n^{c+1} we can compute the $k \times n$ generator matrix G of Enc.

We will use fast matrix multiplication. Specifically, that multiplying an $n \times n^{0.1}$ matrix by an $n^{0.1} \times n$ matrix can be performed in time $O(n^2 \cdot \log^2 n)$ [Cop82] (See [Wil14] for more details on this algorithm). For $i \in [n^{c-1}]$, let $A^{(i)}$ be the $n \times k$ matrix in which the j'th row is $x_{(i-1)n+j}$. Note that the outputs $\mathrm{Enc}(x_1), \dots, \mathrm{Enc}(x_{n^c})$ that we want to compute are the rows of the matrices $A^{(i)} \cdot G$ where i ranges over $[n^{c-1}]$. Each multiplication of $A^{(i)} \cdot G$ takes time $O(n^2 \cdot \log^2 n)$ and therefore n^{c-1} such computations take time $O(n^{c+1} \cdot \log^2 n)$. Overall, the entire computation takes time $O(n^{c+1} \cdot \log^2 n)$, which gives amortized $O(n \cdot \log^2 n)$ time. □

5.3 Proof of Theorem 5.4 and Theorem 5.5

In this section, we put everything together and prove Theorem 5.4 and Theorem 5.5.

[18] We will use η which is only slightly smaller than constant, and so the term $\log(1/\eta)$ is immaterial. We could have been more careful and reduce the running time from $n \cdot \log(1/\eta)$ to expected running time $2n$ by observing that determining whether $\log(1/\eta)$ uniform bits are all one, can be done by querying only two of the bits (in expectation).

Proof of Theorem 5.4. We use the linear code of Theorem 4.1, and apply Lemma 5.9, choosing $\eta = \frac{1}{2 \cdot \log n}$, so that $k/2 + n\log(1/\eta) \leq n \cdot \log n$. The properties in Theorem 5.4 follow directly from Theorem 4.1 and Lemma 5.9, and the fact that $\eta = o(n)$. $\square$

The proof of Theorem 5.5 follows in the same way, by using the following construction of error correcting codes, due to Garcia and Stichtenoth [GS96].

Theorem 5.10. *[GS96] There exist constants $p_{\max} > 0$, $\delta > 0$ and $R > 0$ such that for every sufficiently large m, setting $n = 8^m - 8$, $k = Rn$, there is a binary linear $[n, k]_2$-code that satisfies:*

- *C has a linear encoding map $\mathrm{Enc} : \mathbb{F}_2^k \to \mathbb{F}_2^n$ that runs in time $\mathrm{poly}(n)$.*
- *Enc is decodable from $p_{\max} \cdot n$ errors in time $\mathrm{poly}(n)$.*
- *C has dual distance $\delta \cdot n$.*
- *Moreover, define $\mathrm{Enc}^{\mathrm{trunc}} : \mathbb{F}_2^{k/2} \to \mathbb{F}_2^n$ by $\mathrm{Enc}^{\mathrm{trunc}}(x) = \mathrm{Enc}(x \circ 0^{k/2})$, and consider the linear code $C' = \mathrm{Enc}^{\mathrm{trunc}}(\mathbb{F}_2^{k/2})$. It holds that C' has dual distance $\delta \cdot n$.*

The code of Garcia and Stichtenoth is not a binary code, but rather a code over constant size alphabet. The statement above is obtained by interpreting the code as a binary code. The reader is referred to the appendix of [Shp09] (which was written by Venkat Guruswami) for a precise description of this code and a proof of Theorem 5.10.

6 Stochastic codes for space bounded channels

In this section we state our main construction of stochastic codes for any-order bounded space channels. We start by restating Theorem 1.4 more precisely. The statements below allow a wider range of parameters, and also give a more precise dependence of the parameters on each other.

Theorem 6.1. *There exists a universal constant c_0, such that for every constant $0 \leq p < \frac{1}{2}$, there exists a constant $\delta > 0$, such that for every constant $c_\nu \geq 1$, and every sufficiently small constant $\epsilon > 0$, there exists a constant $L = \mathrm{poly}(1/\epsilon)$, such that for infinitely many N, there is a stochastic code $\mathrm{Enc} : \{0,1\}^{RN} \times \{0,1\}^{O(N \cdot \log N)} \to \{0,1\}^N$ that satisfies the following properties:*

- *Enc has rate $R \geq 1 - H(p) - \epsilon$.*
- *There is a list-decoding algorithm Dec showing that Enc is L-list decodable for any-order space $s = N^\delta$ channels that induce at most pN errors, with probability $1 - \nu$, for $\nu = 2^{-(\log N)^{c_\nu}}$.*
- *Enc can be computed in time $N \cdot (\log N)^{c_0 \cdot c_\nu}$.*
- *Dec can be computed in time $N \cdot (\log N)^{c_0 \cdot c_\nu}$.*

Remark 6.2 (Dependence on constants)**.** *In this remark we give more details on the dependence of the parameters on the chosen constants.*

- *The list size achieved in the proof of Theorem 6.1 is $L = O(\frac{1}{\epsilon^4})$. However, if p is sufficiently smaller than $1/2$ (say $p < 0.49$) then the list size can be reduced to $L = O(\frac{1}{\epsilon^2})$ by a more careful argument, and if p is sufficiently smaller than $1/4$ (say $p < 0.24$) then it can be further reduced to $L = O(1/\epsilon)$.*

- *The running time of encoding and decoding depends on ϵ as follows: For every $\epsilon > 0$ there exists a constant $c_\epsilon = 2^{\mathrm{poly}(1/\epsilon)}$ such that the running time is $c_\epsilon \cdot N \cdot (\log N)^{c_0 \cdot c_\nu}$. This dependence is inherited from the use of explicit codes for binary symmetric channels [For65, GI05]. All other ingredients allow $c_\epsilon = \mathrm{poly}(1/\epsilon)$. Polar codes [GX15, HAU14] achieve running time $c_\epsilon n \log n$, where $c_\epsilon = \mathrm{poly}(1/\epsilon)$. It is plausible that using polar codes (and some other modifications) the dependence on ϵ can be reduced to* $\mathrm{poly}(1/\epsilon)$.
- *The theorem statement does not explicitly state which choices of infinitely many N are possible. Again, the reason that we don't get "for every sufficiently large N" is solely because linear time codes for binary symmetric channels [For65, GI05] are stated for infinitely many N. We remark that an inspection of these results reveals that (in the very least) there exists a universal polynomial $q(\cdot)$ such that for every $\epsilon > 0$, there is a constant c'_ϵ such that for every sufficiently large m, a suitable N can be found between $c'_\epsilon \cdot q(m)$ and $c'_\epsilon \cdot q(m+1)$. The same property is inherited by our construction. We remark that we are less picky and can allow quasilinear time codes for binary symmetric channels, which can be constructed more easily using code concatenation and yield a denser family of N's.*

We can also achieve a different tradeoff where the channel has space $N/\mathrm{polylog}(N)$, for small p, in polynomial time. The following theorem is the more formal restatement of Theorem 1.5.

Theorem 6.3. *There exist universal constants $p_{\max} > 0$ and c_1, such that for every constants $0 \le p \le p_{\max}$, and $c_\nu \ge 1$, and every sufficiently small constant $\epsilon > 0$, there exists a constant $L = \mathrm{poly}(1/\epsilon)$, such that for infinitely many N, there is a stochastic code* $\mathrm{Enc} : \{0,1\}^{RN} \times \{0,1\}^{O(N \cdot \log N)} \to \{0,1\}^N$ *that satisfies the following properties:*

- Enc *has rate $R \ge 1 - H(p) - \epsilon$.*
- *There is a list-decoding algorithm* Dec *showing that* Enc *is L-list decodable for any-order space $s = \frac{N}{(\log N)^{c_1 + c_\nu}}$ channels that induce at most pN error, with probability $1 - \nu$, for $\nu = 2^{-(\log N)^{c_\nu}}$.*
- Enc *can be computed in time* $\mathrm{poly}(N)$.
- Dec *can be computed in time* $\mathrm{poly}(N)$.

In Section 6.1 we present our construction. The construction expects to receive a stochastic control codes with certain properties. In Section 6.2 we plug in the specific control codes of Section 5 to obtain our main results. The correction of the construction is proven in Section 7.

6.1 The construction

In this section we present our construction of stochastic codes for bounded channels. The construction is detailed in three figures. Figure 1 lists parameters and ingredients, Figure 2 which describes the encoding algorithm, and Figure 3 which describes the decoding algorithm. We start with some notation and definitions. We remark that an intuitive explanation of the construction appears in Section 2.2.

Partitioning codewords into control blocks and data blocks. The construction will think of codewords $c \in \{0,1\}^N$ as being composed of $n = n_{\mathrm{ctrl}} + n_{\mathrm{data}}$ blocks of length $b = N/n$. Given a subset $I \subseteq [n]$ of n_{ctrl} distinct indices, we can decompose c into its data part $c_{\mathrm{data}} \in \{0,1\}^{N_{\mathrm{data}} = n_{\mathrm{data}} \cdot b}$ and its control part $c_{\mathrm{ctrl}} \in \{0,1\}^{N_{\mathrm{ctrl}} = n_{\mathrm{ctrl}} \cdot b}$. Similarly, given strings c_{data} and c_{ctrl} we can prepare the codeword c (which we denote by $(c_{\mathrm{data}}, c_{\mathrm{ctrl}})^I$ by the reverse operation. This is stated formally in the definition below.

Definition 6.4. *Let $I = \{i_1, \ldots, i_{n_{\mathrm{ctrl}}}\} \subseteq [n]$ be a subset of indices of size n_{ctrl}.*

Figure 1: Parameters and ingredients for stochastic code

Constants:

- $p > 0$ - The fraction of errors we need to recover from.
- $\epsilon > 0$ - The final code will have rate $R \geq 1 - H(p) - \epsilon$. We assume that $\epsilon > 0$ is sufficiently small compared to p.
- $c_\nu \geq 1$ - We are shooting for a code with success probability $1 - \nu$ for $\nu = 2^{-(\log N)^{c_\nu}}$.

Parameters that are allowed to vary with N:

- N - The length (in bits) of the codeword. Throughout, we assume that N is sufficiently large, and that other parameters are chosen as a function of N. Later choices will also restrict N to be a number of a special form.
- b - We divide the N output bits to $n := N/b$ blocks of length b.
- ℓ - This is the total length of a "control seed". Let $\ell' = \ell/3$. This will be the length of individual "seeds".
- s' - The code will work for any-order space s channels, for $s = s' - (\log N)^{2c_\nu+3}$.

Stochastic control code: The construction receives as a component a stochastic code $\mathrm{Enc}_{\mathrm{ctrl}} : \{0,1\}^\ell \times \{0,1\}^d \to \{0,1\}^b$ such that:

- $\mathrm{Enc}_{\mathrm{ctrl}}$ is $(2^{-s'})$-pseudorandom for any-order space s' ROBPs.
- $\mathrm{Enc}_{\mathrm{ctrl}}$ is L_{ctrl}-list-decodable from $(p + \epsilon) \cdot b$ errors (for some parameter L_{ctrl} to be chosen later) with decoding $\mathrm{Dec}_{\mathrm{ctrl}} : \{0,1\}^b \to (\{0,1\}^\ell)^{L_{\mathrm{ctrl}}}$.

Requirements: $(\log N)^{c_\nu+10} \leq b \leq \frac{N}{(\log N)^{c_\nu+10}}$, $s' \geq (\log N)^{c_\nu+3}$, and $\ell \geq s' \cdot (\log N)^3$.

Some more parameters:

- $\epsilon' = \epsilon/3$ will be the fraction of "control blocks", and set $n_{\mathrm{ctrl}} = \epsilon' \cdot n$, $n_{\mathrm{data}} = n - n_{\mathrm{ctrl}}$.
- Let $N_{\mathrm{ctrl}} = b \cdot n_{\mathrm{ctrl}}$ and $N_{\mathrm{data}} = b \cdot n_{\mathrm{data}}$. (Note that: $n = n_{\mathrm{ctrl}} + n_{\mathrm{data}}$, $N = N_{\mathrm{ctrl}} + N_{\mathrm{data}}$).

Other ingredients that are used:

- Let $\alpha = \frac{\epsilon}{10 \log \frac{1}{p}}$, $p_{\mathrm{BSC}} = p + \alpha$, and $R_{\mathrm{BSC}} = 1 - H(p_{\mathrm{BSC}}) - \epsilon/3$. We apply Theorem 3.20 using $p_{\mathrm{BSC}}, \epsilon/3, N_{\mathrm{data}}$ as choices for p, ϵ, n, respectively. Theorem 3.20 only guarantees the code for infinitely many block lengths, and so we require that $N_{\mathrm{data}} = (1 - \epsilon') \cdot N$ is one of these block lengths. This translates into a restriction on N (which is satisfied for infinitely many N). (See Remark 6.2 for a discussion on the "density" of the block lengths). We obtain an encoding function $\mathrm{Enc}_{\mathrm{BSC}} : \{0,1\}^{R_{\mathrm{BSC}} \cdot N_{\mathrm{data}}} \to \{0,1\}^{N_{\mathrm{data}}}$.
- Let $t = (\log N)^{c_\nu+2}$. We use the $(2^{-10 \cdot t}, t)$-wise permutation $\pi : \{0,1\}^{\ell'} \times [N_{\mathrm{data}}] \to [N_{\mathrm{data}}]$. By Theorem 3.11 we have an explicit construction with seed length $O(t \cdot \log N) \leq \ell'$.
- We use Theorem 3.9 to obtain an $(\frac{\alpha}{100}, \frac{\nu}{N^3})$-sampler with distinct samples $\mathrm{Samp} : \{0,1\}^{\ell'} \to [n]^{n_{\mathrm{ctrl}}}$. By Theorem 3.9 we have an explicit construction with seed length $O(\log \frac{N^3}{\nu}) = O((\log N)^{c_\nu+1}) \leq \ell'$. We use n_{ctrl} samples, and indeed $n_{\mathrm{ctrl}} = \epsilon' \cdot n \geq \epsilon' \cdot (\log N)^{c_\nu+10} \gg \log \frac{N^3}{\nu}$ (as required in Theorem 3.9).
- Let $G : \{0,1\}^{\ell'} \to \{0,1\}^{N_{\mathrm{data}}}$ be the $(2^{-s'})$-PRG for any-order space s' ROBPs, provided by Theorem 3.7. We verify that for the constant c hidden in the statmement of Theorem 3.7, for sufficiently large N, $\ell' \geq c \cdot s' \cdot (\log N_{\mathrm{data}})^2$.

- *Given strings $c_{\mathrm{data}} \in \{0,1\}^{N_{\mathrm{data}}}$ and $c_{\mathrm{ctrl}} \in \{0,1\}^{N_{\mathrm{ctrl}}}$ we define an N bit string c denoted by $(c_{\mathrm{data}}, c_{\mathrm{ctrl}})^I$ as follows: We think of $c_{\mathrm{data}}, c_{\mathrm{ctrl}}, c$ as being composed of blocks of length b (that is*

$c_{\mathrm{data}} \in (\{0,1\}^b)^{n_{\mathrm{data}}}$, $c_{\mathrm{ctrl}} \in (\{0,1\}^b)^{n_{\mathrm{ctrl}}}$ *and* $c \in (\{0,1\}^b)^n$*). We enumerate the indices in* $[n] \setminus I$ *by* $j_1, \ldots, j_{n_{\mathrm{data}}}$ *and set* $c_\ell = \begin{cases} (c_{\mathrm{ctrl}})_k & \text{if } \ell = i_k \text{ for some } k; \\ (c_{\mathrm{data}})_k & \text{if } \ell = j_k \text{ for some } k \end{cases}$

- *Given a string* $c \in \{0,1\}^N$ *(which we think of as* $c \in (\{0,1\}^b)^n$*) we define strings* $c^I_{\mathrm{data}}, c^I_{\mathrm{ctrl}}$ *by* $c^I_{\mathrm{ctrl}} = c|_I$ *and* $c^I_{\mathrm{data}} = c|_{[n]\setminus I}$*, (namely the strings restricted to the indices in I,* $[n] \setminus I$*, respectively).*

We omit the superscript I when it is clear from the context.

Figure 2: Encoding algorithm for stochastic code

Input:

- A message $m \in \{0,1\}^{R_{\mathrm{BSC}} \cdot N_{\mathrm{data}}}$. (This gives $R = \frac{R_{\mathrm{BSC}} \cdot N_{\mathrm{data}}}{N}$).
- A "random part" for the stochastic encoding that consists of: a string $s = (s_{\mathrm{samp}}, s_\pi, s_{\mathrm{PRG}})$ where $s_{\mathrm{samp}}, s_\pi, s_{\mathrm{PRG}} \in \{0,1\}^{\ell'}$ so that $s \in \{0,1\}^\ell$, and $r_1, \ldots, r_{n_{\mathrm{ctrl}}} \in \{0,1\}^d$.

Output: A codeword $c = \mathrm{Enc}(m; (s, r_1, \ldots, r_{n_{\mathrm{ctrl}}}))$ of length N.

Operation:

Determine control blocks: Apply $\mathrm{Samp}(s_{\mathrm{samp}})$ to generate $I = \{i_1, \ldots, i_{n_{\mathrm{ctrl}}}\} \subseteq [n]$. These blocks will be called "control blocks", and the remaining n_{data} blocks will be called "data blocks".

Prepare data part: We prepare a string c_{data} of length N_{data} as follows:

- Encode m by $x = \mathrm{Enc}_{\mathrm{BSC}}(m)$.
- Generate an N_{data} bit string y by reordering the N_{data} bits of the encoding using the (inverse of) the permutation $\pi_{s_\pi}(\cdot) = \pi(s_\pi, \cdot)$. More precisely, $y = \pi^{-1}_{s_\pi}(x) = \pi^{-1}_{s_\pi}(\mathrm{Enc}_{\mathrm{BSC}}(m))$.
- Mask y using PRG. That is, $c_{\mathrm{data}} = y \oplus G(s_{\mathrm{PRG}}) = \pi^{-1}_{s_\pi}(\mathrm{Enc}_{\mathrm{BSC}}(m)) \oplus G(s_{\mathrm{PRG}})$.

Prepare control part: We prepare a string c_{ctrl} of length N_{ctrl} (which we view as n_{ctrl} blocks of length b) as follows:

- $(c_{\mathrm{ctrl}})_j = \mathrm{Enc}_{\mathrm{ctrl}}(s, r_j)$.

Merge data and control parts: We prepare the final output codeword $c \in \{0,1\}^N$ by merging c_{data} and c_{ctrl}. That is, $c = (c_{\mathrm{data}}, c_{\mathrm{ctrl}})^I$.

Theorem 6.5 (correctness of the construction)**.** *There exists a universal constant* c_0 *such that for every constants* $0 \le p < \frac{1}{2}$*,* $c_\nu \ge 1$*, and every sufficiently small constant* $\epsilon > 0$*, for infinitely many* N *we have that: for every* b, ℓ, s'*, and stochastic code* $\mathrm{Enc}_{\mathrm{ctrl}} : \{0,1\}^\ell \times \{0,1\}^d \to \{0,1\}^b$ *that satisfy the requirements in Figure 1. The encoding and decoding functions* $\mathrm{Enc} : \{0,1\}^{Rn} \times \{0,1\}^{\ell + n_{\mathrm{ctrl}} \cdot d} \to \{0,1\}^N$ *and* $\mathrm{Dec} : \{0,1\}^N \to (\{0,1\}^{RN})^{\frac{100 \cdot L_{\mathrm{ctrl}}}{\epsilon^2}}$ *specified in Figures 2 and 3 using the ingredients and parameter choices in Figure 1 satisfy the following properties:*

- Enc *has rate* $R \ge 1 - H(p) - \epsilon$*.*
- Dec *is a list-decoding algorithm showing that* Enc *is* $O(\frac{L_{ctrl}}{\epsilon^2})$*-list decodable for any-order space* s *channels that induce at most* pN *error, with probability* $1 - \nu$*, for* $s = s' - s_A$*,* $s_A \le (\log N)^{2c_\nu + 3}$*, and* $\nu = 2^{-(\log N)^{c_\nu}}$*.*

Figure 3: List-decoding algorithm for stochastic code

Input: A "received word" $c' \in \{0,1\}^N$.
Output: A list of messages $m \in \{0,1\}^{RN}$, where the list is of size at most $\frac{100 \cdot L_{\text{ctrl}}}{\epsilon^2}$.
Operation:

Determine candidates for control information:

Decode control code: Generate $n' = (\log N)^{c_\nu+2}$ lists of size L_{ctrl} as follows: choose uniformly distributed and independent n' indices $i_1, \ldots, i_{n'} \in [n]$, and for every $j \in [n']$ apply the list decoding algorithm, Dec_{ctrl} on c'_{i_j} (here, c'_i is the i'th block of c'). This gives a size L_{ctrl} list, $\text{List}_i = \text{Dec}_{\text{ctrl}}(c'_{i_j})$.

Prune list of candidates: Let $\text{List}_{\text{ctrl}}$ consist of all $s \in \{0,1\}^\ell$ such that $s \in \text{List}_i$ for at least $\frac{\epsilon^2 \cdot n'}{100}$ of $i \in [n']$. Note that $\text{List}_{\text{ctrl}}$ is of size at most $\frac{100 \cdot L_{\text{ctrl}}}{\epsilon^2}$.

Use each control candidate to decode data: For each $s = (s_{\text{samp}}, s_\pi, s_{\text{PRG}}) \in \text{List}_{\text{ctrl}}$ we produce a candidate messages $m_s \in \{0,1\}^{RN}$.

Determine control blocks: Apply $\text{Samp}(s_{\text{samp}})$ to generate $I = \{i_1, \ldots, i_{n_{\text{ctrl}}}\}$. Compute $c'_{\text{data}} = (c')^I_{\text{data}}$.
Unmask PRG: Compute $y'_{\text{data}} = c'_{\text{data}} \oplus G(s_{\text{PRG}})$.
Reverse permutation: Let x' be the N_{data} bit string obtained by "undoing" the permutation. More precisely, let $\pi_{s_\pi}(\cdot) = \pi(s_\pi, \cdot)$, and let $x' = \pi_{s_\pi}(y'_{\text{data}}) = \pi_{s_\pi}(c'_{\text{data}} \oplus G(s_{\text{PRG}}))$.
Decode data: Compute $m_s = \text{Dec}_{\text{BSC}}(x')$.
Prepare output list: The final output is $\text{List} = \{m_s : s \in \text{List}_{\text{ctrl}}\}$.

- Enc *can be computed in time* $N \cdot (\log N)^{c_0 \cdot c_\nu} + T$*, where* T *is a bound on the time it takes to perform the following task: Given* n *pairs* $(m_1, s_1), \ldots, (m_n, s_n) \in \{0,1\}^\ell \times \{0,1\}^d$ *output* $\text{Enc}_{\text{ctrl}}(m_1, s_1), \ldots, \text{Enc}_{\text{ctrl}}(m_n, s_n)$.
- Dec *can be computed in time* $L_{\text{ctrl}} \cdot N \cdot (\log N)^{c_0 \cdot c_\nu} + n' \cdot T'$*, where* $n' = (\log N)^{c_\nu+2}$ *and* T' *is the running time of* Dec_{ctrl} *on input in* $\{0,1\}^b$.

We prove Theorem 6.5 in Section 7. In the next section, we plug in our control codes from Section 5 to get specific constructions.

6.2 Deriving the main theorems

In this section we use specific stochastic control codes to derive our main results. We first use the control code of Theorem 5.4 to prove Theorem 6.1.

Proof. (of Theorem 6.1) We want to choose parameters b, ℓ, s' and a control code to plug into Theorem 6.5. Our plan is to use Theorem 5.4 as a control code. Let $\beta = \frac{1}{2} - p - 2\epsilon$ and note that β is a positive constant as $p < \frac{1}{2}$ is a constant, and $\epsilon > 0$ is sufficiently small. Given $\beta > 0$, Theorem 5.4 provides a constant $0 < \alpha < 0.1$. The running time of encoding and decoding algorithms in Theorem 5.4 is n^c for some universal constant c. Let $\lambda = \frac{1}{2(c+1)}$. We want to choose $b = N^\lambda$ and use it as a block length in Theorem 5.4. However, we must verify that b is of the form $(2^m - 1) \cdot m$ in Theorem 5.4, and so we choose a number b of this form such that $N^\lambda \leq b \leq N^{2\lambda}$ (and such a number exists). We apply Theorem 5.4 to obtain a code $\text{Enc} : \{0,1\}^{b^\alpha} \times \{0,1\}^d \to \{0,1\}^b$ that is $2^{-\hat{s}}$-pseudorandom for any order space $\hat{s}$ ROBPs with $\hat{s} = \frac{b^\alpha}{\log^3 b}$. We also obtain that this code is $O(1/\hat{\epsilon}^2)$-list decodable from $(\frac{1}{2} - \hat{\epsilon})b$ errors in time b^c for every constant $\hat{\epsilon} > \beta$.

We choose $\ell = b^\alpha$ and $s' = \frac{\ell}{\log^3 N} \leq \hat{s}$. It follows that Enc is $2^{-s'}$-pseudorandom for any-order space s' ROBPs. It also follows that Enc is L_{ctrl}-list decodable from $(p+\epsilon)\cdot b$ errors for $L_{\text{ctrl}} = O(\frac{1}{(\frac{1}{2}-(p+\epsilon))^2})$ in time b^c. This follows because $p+\epsilon < p+2\epsilon = \frac{1}{2} - \beta$. This means that $\text{Enc}_{\text{ctrl}} : \{0,1\}^\ell \times \{0,1\}^d \to \{0,1\}^b$ satisfies the requirements from a control code in Figure 1.

Furthermore, the requirements in Figure 1 are met by our choices of b, ℓ and s'. Specifically:

$$(\log N)^{c_\nu+10} \leq N^\lambda \leq b \leq N^{2\lambda} \leq \frac{N}{(\log N)^{c_\nu+10}},$$

$$s' = \frac{\ell}{\log^3 N} = \frac{b^\alpha}{\log^3 N} \geq \frac{N^{\lambda\cdot\alpha}}{\log^3 N} \geq (\log N)^{c_\nu+3},$$

and we chose $s' = \frac{\ell}{\log^3 N}$ so that the requirement $\ell \geq s' \cdot (\log N)^3$ is met. It follows that we meet all the conditions of Theorem 6.5 and obtain that:

- Enc has rate $R \geq 1 - H(p) - \epsilon$.
- Let $L = O(\frac{L_{\text{ctrl}}}{\epsilon^2}) = O(\frac{1}{(\frac{1}{2}-(p+\epsilon))^2\cdot\epsilon^2}) = O(\frac{1}{\epsilon^4})$. There is a list-decoding algorithm Dec showing that Enc is L-list decodable for any-order space $s = s' - s_A \geq N^{\frac{\alpha\cdot\lambda}{2}}$ channels, with probability $1-\nu$. In other words, we can have $\delta = \frac{\alpha\cdot\lambda}{2}$.
- Enc can be computed in time $N \cdot (\log N)^{c_0\cdot c_\nu} + T$ where T is the time it takes to perform $n = N/b$ encodings of Enc_{ctrl}. By Theorem 5.4 there exists a constant c such that performing b^c encodings takes time $O(b^{c+1}\cdot\log^2 b)$. We can break the $n = \frac{N}{b} \geq b^c$ encodings into n/b^c groups of size b^c. Each such group takes time $O(b^{c+1}\log^2 b)$ and so,

$$T = O(\frac{n\cdot b^{c+1}\cdot\log^2 b}{b^c}) = O(n\cdot b\cdot\log^2 b) \leq O(N\cdot\log^2 N).$$

- As ϵ is constant, Dec can be computed in time $N\cdot(\log N)^{c_0\cdot c_\nu} + n'\cdot T'$ where $n' = (\log N)^{c_\nu+2}$ and $T' \leq b^c \leq N^{2\lambda c} \leq N$.

This completes the proof of Theorem 6.1 ☐

We use the control code of Theorem 5.5 to prove Theorem 6.3.

Proof. (of Theorem 6.3) We want to choose parameters b, ℓ, s' and a control code to plug into Theorem 6.5. Our plan is to use Theorem 5.5 as a control code. We want to choose $b = \frac{N}{(\log N)^{c_1+c_\nu}}$ and use it as a block length in Theorem 5.5. However, we must verify that b is of the form $8^m - 8$ in Theorem 5.4, and so we choose a number b of this form such that $\frac{N}{(\log N)^{c_1+c_\nu}} \leq b \leq \frac{N}{(\log N)^{c_1+c_\nu-1}}$ (and such a number exists). We apply Theorem 5.5 to obtain a code $\text{Enc} : \{0,1\}^{R\cdot b} \times \{0,1\}^d \to \{0,1\}^b$ that is $2^{-\hat{s}}$-pseudorandom for any order space $\hat{s}$ ROBPs with $\hat{s} = \frac{b}{\log^2 b}$. We also obtain that this code is decodable from $p_{max}\cdot b$ errors, where $p_{\max} > 0$ is the constant from Theorem 5.5.

We choose $\ell = R \cdot b$ and $s' = \frac{\ell}{\log^3 N} \le \hat{s}$. It follows that Enc is $2^{-s'}$-pseudorandom for any-order space s' ROBPs. This means that $\mathrm{Enc}_{\mathrm{ctrl}} : \{0,1\}^\ell \times \{0,1\}^d \to \{0,1\}^b$ satisfies the requirements from a control code in Figure 1.

Furthermore, the requirements in Figure 1 are met by our choices of b, ℓ and s'. Specifically, for a sufficiently large constant c_1:

$$(\log N)^{c_\nu+10} \le \frac{N}{(\log N)^{c_1+c_\nu}} \le b \le \frac{N}{(\log N)^{c_1+c_\nu-1}} \le \frac{N}{(\log N)^{c_\nu+10}},$$

$$s' = \frac{\ell}{\log^3 N} = \frac{R \cdot b}{\log^3 N} \ge \frac{N}{(\log N)^{c_1+c_\nu} \cdot \log^3 N} \ge (\log N)^{c_\nu+3},$$

and we chose $s' = \frac{\ell}{\log^3 N}$ so that the requirement $\ell \ge s' \cdot (\log N)^3$ is met. It follows that we meet all the conditions of Theorem 6.5 and obtain that:

- Enc has rate $R \ge 1 - H(p) - \epsilon$.
- Let $L = O(\frac{L_{\mathrm{ctrl}}}{\epsilon^2}) = O(\frac{1}{\epsilon^2})$. There is a list-decoding algorithm Dec showing that Enc is L-list decodable for any-order space $s = s' - s_A \ge \frac{N}{(\log N)^{c_1+c_\nu} \cdot \log^3 N}$ channels, with probability $1 - \nu$.
- Enc can be computed in time $\mathrm{poly}(N)$.
- As ϵ is constant, Dec can be computed in time $\mathrm{poly}(N)$.

This completes the proof of Theorem 6.3 $\square$

7 Analyzing the construction

This section is devoted to proving Theorem 6.5, and show the correctness of the main construction.

The setup: Throughout the remainder of the section, we fix the setup of Theorem 6.5. Specifically, let $0 \le p < \frac{1}{2}$, $c_\nu \ge 1$ be constants, and let $\epsilon > 0$ be a sufficiently small constant. Let N be sufficiently large, such that $N_{\mathrm{data}} = (1-\epsilon')N$ is one of the infinitely many block lengths that are guaranteed in Theorem 3.20, as explained in Figure 1. Let b, ℓ, s', be parameters that may depend on N. We also receive a stochastic code $\mathrm{Enc}_{\mathrm{ctrl}} : \{0,1\}^\ell \times \{0,1\}^d \to \{0,1\}^b$, and we assume that all requirements in Figure 1 are satisfied.

Let $\mathrm{Enc} : \{0,1\}^{Rn} \times \{0,1\}^{\ell + n_{\mathrm{ctrl}} \cdot d} \to \{0,1\}^N$ and $\mathrm{Dec} : \{0,1\}^N \to (\{0,1\}^{RN})^{\frac{100 \cdot L_{\mathrm{ctrl}}}{\epsilon^2}}$ be the functions specified in Figures 2 and 3 using the ingredients and parameter choices in Figure 1.

The rate of Enc**.** The rate R of Enc is given by:

$$R = \frac{R_{\mathrm{BSC}} \cdot N_{\mathrm{data}}}{N} = \frac{(1 - H(p_{\mathrm{BSC}}) - \epsilon/3) \cdot (1 - \epsilon') \cdot N}{N} = (1 - H(p + \alpha) - \epsilon/3) \cdot (1 - \epsilon/3).$$

We chose $\alpha = \frac{\epsilon}{10 \log \frac{1}{p}}$, so that $H(p + \alpha) \le H(p) + \epsilon/10$. This holds because the derivative $H'(p)$ is decreasing in the interval $(0, 1)$ and $H'(p) \le \log(1/p)$. This means that $H(p + \alpha) \le H(p) + \alpha \cdot H'(p) \le H(p) + \epsilon/10$. Consequently, we can continue and get:

$$R \ge (1 - H(p) - \epsilon/3 - \epsilon/10) \cdot (1 - \epsilon/3) \ge 1 - H(p) - \epsilon.$$

This proves the first item of Theorem 6.5

The running time of encoding. The encoding algorithm Enc of Figure 2 performs the following tasks:

- It applies the sampler of Theorem 3.9, to get $n_{\text{ctrl}} \le n$ samples. This takes time $n \cdot \text{poly}(\log(N)^{c_\nu}) \le N \cdot \text{poly}(\log(N)^{c_\nu})$.
- It applies the encoding of of Enc_{BSC} from Theorem 3.20. This takes time $O(N_{\text{data}}) = O(N)$.
- It applies the $(2^{-10t}, t)$-wise independent permutation π from Theorem 3.11, $N_{\text{data}} \le N$ times for $t = \text{poly}((\log N)^{c_\nu})$. Each such application takes time $\text{poly}(t \cdot \log N) = \text{poly}((\log N)^{c_\nu})$, and overall, this takes time $N \cdot \text{poly}((\log N)^{c_\nu})$.
- It applies the PRG of Theorem 3.7 to obtain a pseudorandom string of length $N_{\text{data}} \le N$ that is $(2^{-s'})$-pseudorandom for any-order space s' ROBPs. This takes time $N \cdot \text{poly}(\log N)$.
- It applies Enc_{ctrl} on $n_{\text{ctrl}} \le n$ pairs $(s, r_1), \ldots, (s, r_{n_{\text{ctrl}}}) \in \{0,1\}^\ell \times \{0,1\}^d$. This Takes time T.

Overall, for a sufficiently large universal constant c_0, the total running time of Enc is bounded by $N \cdot (\log N)^{c_0 \cdot c_\nu} + T$. This proves the third item of Theorem 6.5.

The running time of decoding. The decoding algorithm Dec of Figure 3 performs the following tasks:

- It applies the decoding algorithm Dec_{ctrl} on n' strings. This takes time $n' \cdot T'$.
- It computes a list of $O(L_{\text{ctrl}}/\epsilon^2) = O(L_{\text{ctrl}})$ candidate control strings. This takes time $n' \cdot L_{\text{ctrl}} \le N \cdot L_{\text{ctrl}}$.
- For each of the $O(L_{\text{ctrl}})$ candidates:
  - It applies the sampler of Theorem 3.9 (with the same parameter used in the encoding) to get t samples where $t = n_{\text{ctrl}} \le n$. This takes time $n \cdot \text{poly}(\log(N)^{c_\nu}) \le N \cdot \text{poly}(\log(N)^{c_\nu})$.
  - It applies the $(2^{-10t}, t)$-wise independent permutation π from Theorem 3.11 $N_{\text{data}} \le N$ times for $t = \text{poly}((\log N)^{c_\nu})$ (same parameters as in encoding). Each such application takes time $\text{poly}(t \cdot \log N) = \text{poly}((\log N)^{c_\nu})$, and overall, this takes time $N \cdot \text{poly}((\log N)^{c_\nu})$.
  - It applies the PRG of Theorem 3.7 to obtain a pseudorandom string of length $N_{\text{data}} \le N$ that is $(2^{-s'})$-pseudorandom for any-order space s' ROBPs (same parameters as in encoding). This takes time $N \cdot \text{poly}(\log N)$.
  - It applies the decoding of Dec_{BSC} from Theorem 3.20. This takes time $O(N_{\text{data}}) = O(N)$.

Overall, for a sufficiently large universal constant c_0, the total running time of Enc is bounded by $L_{\text{ctrl}} \cdot N \cdot (\log N)^{c_0 \cdot c_\nu} + n' \cdot T'$. This proves the fourth item of Theorem 6.5.

Milestones for correct decoding. We need to show that for any bounded space channel, the output list of the decoding algorithm contains the encoded message with high probability. Following Guruswami and Smith [GS16], and Shaltiel and Silbak [SS16] we will analyze the construction in two steps:

- We first consider the case that the channel e^σ_C is an "additive channel", namely that $e^\sigma_C(z) = e$ for some fixed error vector e.
- At the second step we extend to general channels that can choose the error pattern as a function of z. The second step is done using a reduction to the first step.

The high level idea is to show that for certain events "that we will call milestones", we can use the pseudorandomness properties of the construction to show that these milestones "do not distinguish" between the first setup and the second setup.

Following [SS16], we present the following abstraction of this method: We will define "milestones" (as a function of m, s_π, s_{samp} and e) and will require that:

1. If the milestones occur, then the decoding algorithm succeeds w.h.p.
2. If S_π, S_{samp} are random and e is fixed (that is, if the channel is additive) then the milestones occur with probability close to one.
3. Checking whether the milestones occur is feasible for small space ROBPs.

We will state a general lemma showing that if such milestones exist, then the correctness of the decoding holds even against general any-order small space channels (that are not necessarily additive).

Before proceeding with the formal definition, let us give some high level intuition for why this approach works: We start by considering an additive channel $e_C^\sigma(z) = e$ which chooses the error pattern e without looking at the codeword z. The second property says that in this setting, the milestones occur with probability close to one. The first property says that that whenever the milestone occur, decoding is successful. Thus, the milestones establish the correctness of the construction for an additive channel. Now, let us consider a general channel $e_C^\sigma(z) = e$ that chooses its error pattern e as a function of the transmitted codeword z. We have set up the construction so that we can argue that for every fixing of m, s_π, s_{samp}, the transmitted codeword z is pseudorandom for the channel. Thus, when the milestones function is applied on the error pattern e, it cannot distinguish the case where the channel sees the real codeword z, from the case where it sees an independent uniform string. In the latter case, the behavior of the channel is *additive*, as it chooses its errors in a way that is independent of the codeword, and we have already argued that decoding succeeds in this scenario. It follows that decoding succeeds w.h.p. even in the general scenario where the channel chooses its error as a function of the codeword.[19]

The precise properties of a milestones functions are stated formally in the definition and theorem below. In the definition we will allow the milestones function to also toss random coins (denoted by y).

Definition 7.1 (Milestones function)**.** *Let* $A : \{0,1\}^{RN} \times \{0,1\}^{\ell'} \times \{0,1\}^{\ell'} \times \{0,1\}^N \times \{0,1\}^N \to \{0,1\}$ *be a function that receives as input:*

- *A message* $m \in \{0,1\}^{RN}$.
- *A sampler seed* $s_{\text{samp}} \in \{0,1\}^{\ell'}$.
- *A permutation seed* $s_\pi \in \{0,1\}^{\ell'}$.
- *An error vector* $e \in \{0,1\}^N$ *of hamming weight at most* pN.
- *A choice for "random coins"* $y \in \{0,1\}^N$.

We say that A *is a* ***milestones function*** *if it has all the following properties: (the probability space for the statements below is choosing the randomness of the encoder* $S = (S_{\text{samp}}, S_\pi, S_{PRG}) \in \{0,1\}^\ell$, $R = (R_1, \dots, R_{n_{\text{ctrl}}}) \in (\{0,1\}^d)^{n_{\text{ctrl}}}$ *and* Y *(the coins of* A*) uniformly and independently.)*

[19] We stress that the third requirement from a milestones function is that checking whether the event occurred can be done by bounded space ROBPs. This is a very stringent requirement, as the steps run by the decoding algorithm (like decoding using Dec_{ctrl} or Dec_{BSC}) are not computable by small space ROBPs. We need to come up with an event that on the one hand is easy to check, and on the other hand implies that decoding succeeds. This means that there is an inherent conflict between the first and third requirements of a milestones function, and indeed the choice of our final milestones function is somewhat delicate.

1. *For every $m \in \{0,1\}^{RN}$, $s \in \{0,1\}^{\ell}, r \in (\{0,1\}^d)^{n_{\text{ctrl}}}$ and $e \in \{0,1\}^n$ of Hamming weight at most pN: If $\Pr[A(m, s_{\text{samp}}, s_\pi, e, Y) = 1] \geq \frac{1}{2}$ then the probability (over the random coins of* Dec*) that $m \in \text{Dec}(\text{Enc}(m;(s,r)) \oplus e)$ is at least $1 - \frac{\nu}{20}$.*
2. *For every $m \in \{0,1\}^{RN}$ and $e \in \{0,1\}^n$ of Hamming weight at most pN, $\Pr[A(m, S_{\text{samp}}, S_\pi, e, Y) = 1] \geq 1 - \frac{\nu}{20}$.*
3. *For every $m, s_{\text{samp}}, s_\pi, y$ and every any-order space s channel e_C^σ that induces at most pN errors, the function $D(z) = A(m, s_{\text{samp}}, s_\pi, e_C^\sigma(z), y)$ has a space s' ROBP F such that $D(z) = F^\sigma(z)$.*

Lemma 7.2 (Milestones Lemma)**.** *If there exist a milestones function then for every $m \in \{0,1\}^{RN}$, and every any-order space s channel e_C^σ,*

$$\Pr[m \in \text{Dec}(\text{Enc}(m,S,R) \oplus e_C^\sigma(\text{Enc}(m,S,R)))] \geq 1 - \nu$$

To conclude the proof of Theorem 6.5 and prove its second item we need to:

- Prove Lemma 7.2. This is done in Section 7.1
- Provide a milestones function A for our construction. This is done in Section 7.2.

Together, these two tasks conclude the proof of Theorem 6.5.

7.1 Proof of Milestones Lemma

We prove the milestones lemma in two steps, described in the two sections below. The proof follows along the same lines as in [SS16] which in turn relies on [GS16].

7.1.1 The hiding lemma

The following lemma states that for every message, sampler seed and permutation seed, The encoding is pseudorandom for small space any-order ROBPs. This intuitively means that from the point of view of small space channel, the codeword that it sees is independent of the choices of the message, sampler seed and permutation seed. This can be used to argue that (in some precise sense explained later) the errors inflicted by such a channel are independent of the message, sampler seed, and permutation seed. This will later allow us to analyze the channel as if the error pattern it chooses is independent of the message, sampler seed and permutation seed.

Lemma 7.3 (Hiding Lemma)**.** *For every message $m \in \{0,1\}^{RN}$, sampler seed $s_{\text{samp}} \in \{0,1\}^{\ell'}$ and permutation seed $s_\pi \in \{0,1\}^{\ell'}$, let $V = \text{Enc}(m; (s_\pi, s_{samp}, S_{PRG}, R_1, \cdots, R_{n_{ctrl}}))$ be a random variable (defined over the probability space where $S_{PRG}, R_1, \cdots, R_{n_{ctrl}}$ are chosen uniformly and independently). It follows that V is $\frac{\nu}{5}$-pseudorandom for any-order space s' ROBPs.*

Proof. We assume for contradiction that there exists a space s' ROBP D and a permutation $\sigma \colon \{0,1\}^N \to \{0,1\}^N$ such that:

$$|\Pr[D^\sigma(V) = 1] - \Pr[D^\sigma(U_N) = 1]| > \frac{\nu}{5}$$

The lemma follows from the following claim.

Claim 7.4. *One of the following holds:*

- *There exists a space s' ROBP $C : \{0,1\}^{N_{\text{data}}} \to \{0,1\}$ and a permutation $\tau : [N_{\text{data}}] \to [N_{\text{data}}]$ such that,* $|\Pr[C^\tau(G(S_{\text{PRG}})) = 1] - \Pr[C^\tau(U_{N_{\text{data}}}) = 1]| > \frac{\nu}{10}$.
- *There exists $z' \in \{0,1\}^\ell$ and space s' ROBP $C : \{0,1\}^b \to \{0,1\}$ and a permutation $\tau : [b] \to [b]$ such that,* $|\Pr[C^\tau(\text{Enc}_{\text{ctrl}}(z', U_d)) = 1] - \Pr[C^\tau(U_b) = 1]| > \frac{\nu}{10n}$.

Proof. (of claim) We partition V into $V = (V_{\text{data}}, V_{\text{ctrl}})^{\text{Samp}(s_{\text{samp}})}$ using definition 6.4. We have that D^σ distinguishes $V = (V_{\text{data}}, V_{\text{ctrl}})$ from $U_N = (U_{\text{data}}, U_{\text{ctrl}})$ with probability greater than $\nu/5$, we do a hybrid argument and consider the hybrid distribution $H = (V_{\text{data}}, U_{\text{ctrl}})$. It follows that:

- Either D^σ distinguishes H from U_N with probability $\nu/10$,
- or, D^σ distinguishes H from V with probability $\nu/10$.

In the first case, we have that V_{data} and U_{ctrl} are independent, and an averaging argument gives that there exists a fixed value v'_{ctrl}, such that D^σ distinguishes $(U_{\text{data}}, v'_{\text{ctrl}})$ from $(V_{\text{data}}, v'_{\text{ctrl}})$ with probability $\nu/10$. This gives that there exists a space s' ROBP $C : \{0,1\}^{N_{\text{data}}} \to \{0,1\}$ and a permutation $\tau : [N_{\text{data}}] \to [N_{\text{data}}]$ such that the first item of the claim holds.

In the second case, we have that m and s_π are fixed and therefore the string $y = \pi_{s_\pi}^{-1}(\text{Enc}_{\text{BSC}}(m))$ used in the encoding algorithm is also fixed. The encoding algorithm computes the data part by xoring y with $G(S_{\text{PRG}})$ and therefore $V_{\text{data}} = G(S_{\text{PRG}}) \oplus y$. By an averaging argument, there exists a fixing s'_{PRG} such that D^σ distinguishes $((G(s'_{\text{PRG}}) \oplus y), U_{\text{ctrl}})$ from $(((G(s'_{\text{PRG}}) \oplus y), V_{\text{ctrl}})|S_{\text{PRG}} = s'_{\text{PRG}})$ with probability $\nu/10$.

We get that there exists a space s' ROBP $D' : \{0,1\}^{n_{\text{ctrl}} \cdot d} \to \{0,1\}$ and a permutation $\sigma' : [n_{\text{ctrl}} \cdot d] \to [n_{\text{ctrl}} \cdot d]$ such that $(D')^{\sigma'}$ distinguishes U_{ctrl} from $V'_{\text{ctrl}} = (V_{\text{ctrl}}|S_{PRG} = s'_{\text{PRG}})$.

Recall that the encoding procedure prepares the j'th block of the control part c_{ctrl}, by $\text{Enc}_{\text{ctrl}}(s, r_j)$.

Having fixed $S_{\text{PRG}} = s'_{\text{PRG}}$ the only random variables that remain unfixed in V'_{ctrl} are $R_1, \ldots, R_{n_{\text{ctrl}}}$. This means that there exists $s' \in \{0,1\}^\ell$ such that $(V'_{\text{ctrl}})_j = \text{Enc}_{\text{ctrl}}(s', R_j)$ and in particular, the n_{ctrl} blocks are independent. We have that $(D')^{\sigma'}$ distinguishes V'_{ctrl} from U_{ctrl} with probability $\nu/10$, and by a standard hybrid argument, there exists a space s' ROBP C and a permutation $\tau : [b] \to [b]$ such that C^τ distinguishes $(V'_{\text{ctrl}})_j = \text{Enc}_{\text{ctrl}}(s', R_j)$ from uniform with probability $\frac{(\nu/10)}{n_{\text{ctrl}}} \geq \frac{\nu}{10n}$ and the second item follows. ☐

The lemma follows by the pseudorandomness properties of the G and Enc_{ctrl}, noting that

$$\frac{\nu}{10n} \geq \frac{1}{2^{(\log N)^{(c_\nu+1)}}} \geq 2^{-s'}.$$

where the last inequality follows from the requirement that $s' \geq (\log N)^{(c_\nu+3)}$. ☐

7.1.2 Hiding lemma implies milestones lemma

We now show that the milestones lemma (Lemma 7.2) follows from the hiding lemma (Lemma 7.3). We are assuming that A is a milestone function. We need to show that for every message $m \in \{0,1\}^{RN}$, and every any-order space s channel e_C^σ that induces at most pN errors,

$$\Pr[m \in \text{Dec}(\text{Enc}(m, S, R) \oplus e_C^\sigma(\text{Enc}(m, S, R)))] \geq 1 - \nu,$$

where $S = (S_{\text{samp}}, S_\pi, S_{PRG})$, $R = (R_1, \ldots, R_{n_{\text{ctrl}}})$ are chosen uniformly and independently.

Fix some message $m \in \{0,1\}^{RN}$ and let $Z = \text{Enc}(m, S, R)$ denote the random variable that is the encoding of the message. We assume (for contradiction) that $\Pr[m \in \text{Dec}(\text{Enc}(m, S, R) \oplus e_C^\sigma(\text{Enc}(m, S, R)))] < 1 - \nu$. By the first property of a milestones function and an averaging argument we have that:

Claim 7.5. $\Pr[A(m, S_{\mathrm{samp}}, S_\pi, e_C^\sigma(Z), Y) = 1] < 1 - \nu/3$.

Proof. Let

$$B = \{(s, r) | \Pr[m \in \mathrm{Dec}(\mathrm{Enc}(m, s, r) \oplus e_C^\sigma(\mathrm{Enc}(m, s, r)))] < 1 - \nu/20\},$$

where the probability is only over the random coins of the decoder Dec (as m, s, r are fixed). By an averaging argument, $\Pr[(S, R) \in B] \geq \nu - \nu/20$.

Note that for a fixed (s, r) the error vector e induced by the channel C is also fixed. We consider the probability space where $(S, R) = (s, r)$ are fixed and Y (the random coins of the function A) is chosen uniformly. By the first property of a milestone function, we have that for a fixed $(s, r) \in B$ and a fixed error e, $\Pr[A(m, s_{\mathrm{samp}}, s_\pi, e, Y) = 0] > \frac{1}{2}$ (as otherwise decoding must succeed with probability $1 - \nu/20$). Let $A' = A(m, S_{\mathrm{samp}}, S_\pi, e_C^\sigma(Z), Y)$ be the random variable of the output of function A in the probability space where S, R, Y are chosen uniformly.

$$\Pr[A' = 0] \geq \Pr[A' = 0 | (S, R) \in B] \cdot \Pr[(S, R) \in B] > \frac{1}{2} \cdot \frac{19\nu}{20} \geq \frac{\nu}{3}$$

It follows that

$$\Pr[A(m, S_{\mathrm{samp}}, S_\pi, e_C^\sigma(Z), Y) = 1] = \Pr[A' = 1] = 1 - [A' = 0] < 1 - \nu/3.$$

$\square$

We add an independent random variable Z_U that is uniform over $\{0, 1\}^N$ to our probability space (that now consists of independently chosen S, R, Y, Z_U). By the second property of a milestone function, we have that for every error vector e,

$$\Pr[A(m, S_{\mathrm{samp}}, S_\pi, e, Y) = 1] \geq 1 - \nu/20.$$

As Z_U is independent of $(S_{\mathrm{samp}}, S_\pi, Y)$ this holds also for an error vector of the form $e_C^\sigma(Z_U)$. Namely,

$$\Pr[A(m, S_{\mathrm{samp}}, S_\pi, e_C^\sigma(Z_U), Y) = 1] \geq 1 - \nu/20.$$

This means that:

$$\begin{aligned} &\Pr[A(m, S_{\mathrm{samp}}, S_\pi, e_C^\sigma(Z_U), Y) = 1] - \Pr[A(m, S_{\mathrm{samp}}, S_\pi, e_C^\sigma(Z), Y) = 1] \\ &> (1 - \nu/20) - (1 - \nu/3) \geq \nu/4. \end{aligned}$$

By averaging, there exist fixed values $s'_{\mathrm{samp}}, s'_\pi$ and y' such that if we consider the event

$$W = \left\{S_{\mathrm{samp}} = s'_{\mathrm{samp}}, S_\pi = s'_\pi, Y = y'\right\}$$

we have that:

$$\Pr[A(m, s'_{\mathrm{samp}}, s'_\pi, e_C^\sigma(Z_U), y') = 1 | W] - \Pr[A(m, s'_{\mathrm{samp}}, s'_\pi, e_C^\sigma(Z), y') = 1 | W] > \nu/4.$$

We have that (S_{samp}, S_π, Y) is independent of Z_U and also independent of (S_{PRG}, R). Therefore:

$$\Pr[A(m, s'_{\mathrm{samp}}, s'_\pi, e_C^\sigma(Z_U), y') = 1] - \Pr[A(m, s'_{\mathrm{samp}}, s'_\pi, e_C^\sigma(\mathrm{Enc}(m, s'_\pi, s'_{\mathrm{samp}}, S_{PRG}, R)), y') = 1] > \nu/4.$$

This setup (namely, where S_{samp}, S_π are fixed, and $S_{PRG}, R = (R_1, \ldots, R_{n_{\text{ctrl}}})$ are uniform) is exactly the probability space considered in the hiding lemma (Lemma 7.3). By the third property of milestones functions, we have the function $D(z) = A(m, s'_{\text{samp}}, s'_\pi, e^\sigma_C(z), y')$ has a space s'-ROBP F such that $D(z) = F^\sigma(z)$. Therefore,

$$\Pr[F^\sigma(Z_U) = 1] - \Pr[F^\sigma(\text{Enc}(m, s'_\pi, s'_{\text{samp}}, S_{PRG}, R)) = 1] > \nu/4,$$

and this a contradiction to the hiding lemma (Lemma 7.3). This concludes the proof of the milestones lemma.

7.2 Milestones Lemma implies Theorem 6.5

In this section we show that Lemma 7.2 implies Theorem 6.5. Our task is to define a milestones function that meets the three requirements in Definition 7.1.

7.2.1 Intuition for the proof

Our milestones function will be a conjunction of many "small milestones" that:

- If all of them happen then decoding succeeds w.h.p. (so that we meet the first requirement of a milestone function).
- The probability that all milestones occur simultaneously is very large (so that we meet the second requirement of a milestone function).

We will start with defining milestones that are not necessarily computable by small space ROBPs, and will later show that these milestones can be "approximated" by a small space ROBP, so that we meet the third requirement of a milestone function.

A milestone function receives (amongst other things) s_{samp}, s_π and $e \in \{0,1\}^N$ with hamming weight at most pN. We will use the following notation (whenever $s_{\text{Samp}}, s_\pi, e$ are clear from the context).

- $I = \{i_1, \ldots, i_{n_{\text{ctrl}}}\} = \text{Samp}(s_{\text{samp}})$ be the set of control indices.
- $e_{\text{data}} = e^I_{\text{data}}$ and $e_{\text{ctrl}} = e^I_{\text{ctrl}}$ is the partition of the error vector to its control and data part.
- $e^\pi_{\text{data}} = \pi_{s_\pi}(e_{\text{data}})$. This is the permuated error vector on the data part.
- *blocks*$(q) = |\left\{j \in [n_{\text{ctrl}}] : \text{ The hamming weight of } e_{i_j} \text{ is larger than } q \cdot b\right\}|$. (Here e_i denotes the b bit long i'th block of e). This is the number of control blocks on which the fraction of errors is larger than q.

Loosely speaking, the milestones that we will be interested in are:

Control milestone: That *blocks*$(p + \epsilon) \le (1 - \frac{\epsilon}{16}) \cdot n_{\text{ctrl}}$. This means that at least an $\epsilon/16$ fraction of the original n_{ctrl} control blocks were not hit by too large error. (This will happen w.h.p. by the properties of the sampler).

Data milestone: Here we recall the function A_m from Theorem 3.20, and note that our milestones can depend on m. This function $A_m(\cdot)$ receives a string $e' \in \{0,1\}^{N_{\text{data}}}$ and if it answers one, then (amongst other things) $\text{Dec}_{\text{BSC}}(\text{Enc}_{\text{BSC}}(m) \oplus e') = m$. Furthermore Theorem 3.20 guarantees that if e' is obtained by applying a t-wise independent permutation π on e_{data} then $\Pr[A_m(e') = 1]$ is very large.

The final milestones function is the conjunction of both control and data milestones. Continuing this intuitive explanation:

- If both milestones pass, then decoding succeeds w.h.p.: This is because at least $\epsilon^2 n_{\text{ctrl}}/16$ control blocks are decoded correctly by Enc_{ctrl}. This means that the "correct seed s" appears w.h.p. in the list of candidates of the decoding. When decoding the data part is executed for the correct s, the error pattern e' that is encountered is indeed the application of $\pi(e_{\text{data}})$ for a low weight e_{data}, and therefore applying BSC decoding (after unmasking and permuting) indeed recovers the correct message m.
- The probability that both milestones occur simultaneously is very close to one.

A technical difficulty is that this choice of milestones cannot be computed by a small space ROBP and does not meet the third requirement. The issue is that the channel gets to choose the permutation σ that reorders the N bit codeword that it reads. The new order does not necessarily "respect the block structure": The ROBP may read different bits of a certain block c_i' in different times. Moreover, the order in which the ROBP reads the data bits is also modified by the permutation π.

We will use the following approach to deal with this problem. We will consider two versions of a milestones function: a *strong* version and a *weak* version (in which we use different choices of parameters). We will argue that:

- The strong version implies the weak version.
- If the weak version passes then decoding succeeds w.h.p.
- The probability that the strong version passes is close to one.
- This shows that both weak and strong versions satisfy the first two conditions of a milestones function. The importance of this is that any function that is "sandwiched" between these two milestones functions also satisfies the first two conditions of a milestones function. We will implement such a function by a small space ROBP (meeting also the third condition). Specifically:
- For every $m, s_{\text{samp}}, s_\pi, e$ and every permutation $\sigma : [N] \to [N]$ we will show that there is a randomized small space ROBP F such that:
  - If all the strong versions hold with respect to $m, s_{\text{samp}}, s_\pi, e$ then $\Pr_Y[F_Y^\sigma(e) \text{ accepts}] \geq 1 - \nu/20$. (Here the probability is with respect to a random coin toss Y of the milestone function/ROBP).
  - If $\Pr[F_Y^\sigma \text{ accepts}] \geq \frac{1}{2}$ then all the weak version hold with respect to $m, s_{\text{samp}}, s_\pi, e$.

This will gives a space s ROBP that satisfies the third property of a milestone function, and is sandwiched between the weak and strong conditions. Altogether, this will allow us to meet all three conditions. This intuition is implemented in the next sections.

7.2.2 Weak and strong milestone functions

We start by defining weak and strong versions of milestones.

Definition 7.6. *Let $x = (m, s_{\text{samp}}, s_\pi, e, y)$ be an input to a milestone function.*

Control milestone: *We define:*

- $A_{\text{ctrl}}^{weak}(x) = 1$ *iff* blocks$(p + \epsilon) \leq (1 - \frac{\epsilon}{16}) \cdot n_{\text{ctrl}}$.

- $A^{strong}_{\text{ctrl}}(x) = 1$ *iff* **blocks**$(p + \frac{\epsilon}{4}) \leq (1 - \frac{\epsilon}{3}) \cdot n_{\text{ctrl}}$.

Data milestone: *Recall that the code* $\text{Enc}_{\text{BSC}} : \{0,1\}^{R_{\text{BSC}} N_{\text{data}}} \to \{0,1\}^{N_{\text{data}}}$ *that we use is given by Theorem 3.20 using* $p_{\text{BSC}} = p + \alpha$. *Furthermore, Theorem 3.20 also states that this code is a concatenated code with certain additional properties. Using the notation of Theorem 3.20, we define* $A^m_v : \{0,1\}^{N_{\text{data}}} \to \{0,1\}$ *be the function that outputs one on input* $e' \in \{0,1\}^{N_{\text{data}}}$ *iff*

$$|\left\{i \in [n_{\text{out}}] : \text{Dec}^i_{\text{in}}(\text{Enc}(m) \oplus e') \neq \text{Enc}_{\text{out}}(m)_i\right\}| \leq v.$$

(With this notation the function A_m *defined in Theorem 3.20 is* $A^m_{w/10}$*). We define:*

- $A^{weak}_{\text{data}}(x) = 1$ *iff* $A^m_w(e^\pi_{\text{data}}) = 1$.
- $A^{strong}_{\text{data}}(x) = 1$ *iff* $A^m_{w/10}(e^\pi_{\text{data}}) = 1$.

Let $A^{weak}(x) = A^{weak}_{\text{ctrl}}(x) \wedge A^{weak}_{\text{data}}(x)$ *and* $A^{strong}(x) = A^{strong}_{\text{ctrl}}(x) \wedge A^{strong}_{\text{data}}(x)$.

The next two lemmas give that any milestone function that is "sandwiched" between A^{weak} and A^{strong} satisfy the first two properties of a milestone function.

Lemma 7.7. *The function* A^{weak} *satisfies the first property of a milestone function. (This in particular implies that* A^{strong} *also satisfies the first property).*

This follows as the function A^{weak} was defined precisely so that the decoding components, in the decoding algorithm of Figure 3 are used with the correct guarantee. A full proof appears in Section 7.2.4.

Lemma 7.8. *The function* A^{strong} *satisfies the second property of a milestone function (even if* $\nu/20$ *is replaced with* $\nu/100$*). (This in particular implies that* A^{weak} *also satisfies the second property).*

This follows as the function A^{strong} was defined precisely so that the pseudorandom components (the sampler and permutation) are "sufficiently random" to imply that A^{strong} holds. For this, we only need to analyze the case where e is fixed and the seeds (S_{samp}, S_π) are chosen at random. A full proof appears in Section 7.2.5.

7.2.3 Efficient milestones that are sandwiched between weak and strong

Our goal is to define a function $A(m, s_{\text{samp}}, s_\pi, e, y)$ that satisfies all three properties of a milestones function in Definition 7.1. The two functions A^{strong}, A^{weak} of the previous section satisfied the first two properties but not the third (and they also didn't make use of their ability to toss random coins y).

The third condition says that for every $m, s_{\text{samp}}, s_\pi, y$ and every any-order space s channel e^σ_C that induces at most pN errors, the function $D(z) = A(m, s_{\text{samp}}, s_\pi, e^\sigma_C(z), y)$ has a space s' ROBP F such that $D(z) = F^\sigma(z)$.

Let us consider the following computational model for computing $A(m, s_{\text{samp}}, s_\pi, e, y)$.

- The milestone function A has space s_A (for some parameter s_A).
- Computation on the internal space (and all inputs other than e) is for free.
- The milestone function A accesses e in the following manner: A reads the bits of e one by one in an unknown order. However, whenever a bit is read, the milestone function is informed what is the index j of this bit in e.

It follows that if A can be computed in the manner described above, with space $s_A = s' - s$, then the third condition holds and, for every $m, s_{\text{samp}}, s_\pi, y$ and every any-order space s channel e^σ_C that induces at most pN errors, the function $D(z) = A(m, s_{\text{samp}}, s_\pi, e^\sigma_C(z), y)$ has a space s' ROBP F such that $D(z) = F^\sigma(z)$.

More precisely, when C^σ is applied on an input z, it produces the output $C^\sigma(z)$ which is a reordering of $e^\sigma_C(z)$ according to some fixed permutation (the permutation σ^{-1}). The model of computation for A doesn't care about the precise order in which $e = e^\sigma_C(z)$ is given, and therefore the overall computation $D(z) = A(m, s_{\text{samp}}, s_\pi, e^\sigma_C(z), y)$ can be implemented by a space $s_A + s = s'$ ROBP F that reads its input in the order dictated by the permutation σ (that was chosen by the channel).

We now define the final milestones function A that we will use:

Definition 7.9. *Let $x = (m, s_{\text{samp}}, s_\pi, e, y)$ be an input to a milestone function. We will think of y as a pair $y = (y_1, y_2)$ of choices for random coins. We set $q = (\log N)^{c_\nu+1}$.*

Control milestone: *We define $A_{\text{ctrl}}(x)$ as follows:*

- *We think of y_1 as "random coins" for:*
  - *Selecting q uniform and i.i.d. "control blocks" $d^1, \dots, d^q$ from $I = \text{samp}(s_{\text{samp}}) = \{i_1, \dots, i_{n_{\text{ctrl}}}\}$.*
  - *Let $B_i = \{(i-1)b+1, \dots, ib\}$ be the set of indices in the i'th block. For every $k \in [q]$, we also choose uniform and independent $d^k_1, \dots d^k_q \in B_{d^k}$.*
- *$A_{\text{ctrl}}(x) = 1$ iff at least $\frac{\epsilon}{8} \cdot q$ of $k \in [q]$, satisfy that at most $(p + \frac{\epsilon}{2}) \cdot q$ of $j \in [q]$ have $e_{d^k_j} = 1$.*

Data milestone: *We define $A_{\text{ctrl}}(x)$ as follows:*

- *We recall (once again) that the code $\text{Enc}_{\text{BSC}} : \{0,1\}^{R_{\text{BSC}} N_{\text{data}}} \to \{0,1\}^{N_{\text{data}}}$ that we use is given by Theorem 3.20 using $p_{\text{BSC}} = p + \alpha$. Furthermore, Theorem 3.20 also states that this code is a concatenated code with certain additional properties. We write $N_{\text{data}} = n_{\text{out}} \cdot n_{\text{in}}$ as done in Theorem 3.20.*
- *We think of y_2 as "random coins" selecting q uniform and i.i.d. "inner blocks" $v^1, \dots, v^q$ from $[n_{\text{out}}]$.*
- *$A_{\text{data}}(x) = 1$ iff at least $\frac{w}{5} \cdot q$ of $k \in [q]$, satisfy that $\text{Enc}_{\text{out}}(m)_{v^k} = \text{Dec}^{v_k}_{\text{in}}(\text{Enc}_{\text{BSC}}(m) \oplus \pi_{s_\pi}(e_{\text{data}}))$.*

Let $A(x) = A_{\text{ctrl}}(x) \wedge A_{\text{data}}(x)$.

Note that A_{ctrl} can be implemented with space q^2 in the computational model defined above. This is because A_{ctrl} only depends on q^2 specific bits of e. We can store these bits as they arrive, and produce the output when all these bits are stored in memory. Similarly, A_{data} can be implemented with space $q \cdot n_{\text{in}}$ in the computational model defined above. This is because A_{ctrl} only depends on $q \cdot n_{\text{in}}$ specific bits of e.

Overall, this uses space $s_A = q^2 + q \cdot n_{\text{in}} = O(q^2)$ (as n_{in} is a constant). We have chosen $q = (\log N)^{c_\nu+1}$, and so, $s' - s = s_A \le (\log N)^{2c_\nu+3}$ as required in Theorem 6.5.

Loosely speaking, the function A uses its random coins to approximate the functions A^{weak}, A^{strong}. Consequently, by a Chernoff bound we can prove the next lemma. Recall that A^{strong}, A^{weak} do not depend on the input y, and so in the lemma below we allow ourselves to omit it from their input.

Lemma 7.10. *For every $m \in \{0,1\}^{RN}$, $s_{\text{samp}} \in \{0,1\}^{\ell'}$, $s_\pi \in \{0,1\}^{\ell'}$ and $e \in \{0,1\}^N$ of hamming weight at most pN:*

- *If* $\Pr_Y[A(m, s_{\text{samp}}, s_\pi, e, Y) = 1] \geq \frac{1}{2}$ *then* $A^{weak}(m, s_{\text{samp}}, s_\pi, e) = 1$.
- *If* $A^{strong}(m, s_{\text{samp}}, s_\pi, e) = 1$ *then* $\Pr_Y[A(m, s_{\text{samp}}, s_\pi, e, Y) = 1] \geq 1 - \nu/100$.

Loosely speaking, the claim follows because A approximates a function that is sandwiched between A^{weak} and A^{strong}.

Proof. We start with proving the second item, and consider the control milestone. For every control block $i \in I$, let p_i be the relative hamming weight of e_i. If A^{strong} accepts then *blocks*$(p + \frac{\epsilon}{4}) \leq 1 - \frac{\epsilon}{3}$. This means that the fraction of control blocks in $i \in I$ such that $p_i > p + \frac{\epsilon}{4}$ is at most $1 - \frac{\epsilon}{3}$. By a Chernoff bound, the probability that more than an $\epsilon/4$ fraction of the q selected $d^1, \ldots, d^q$ have $p_i > p + \frac{\epsilon}{4}$ is $2^{-\Omega(q/\epsilon^2)}$. By another Chernoff bound for every $k \in [q]$, the probability (over the choice of $d_1^k, \ldots, d_q^k$) that

$$\Pr\left[\ |\ \frac{1}{q} \cdot \sum_{j \in [q]} e_{d_j^k} - p_i \ |> \frac{\epsilon}{10}\right] \leq e^{-\Omega(q/\epsilon^2)}.$$

Therefore, by a union bound, with probability $1 - q \cdot e^{-\Omega(q/\epsilon^2)}$ all approximations are correct and therefore, with probability $1 - (q+1) \cdot 2^{-\Omega(q/\epsilon^2)} \geq 1 - \nu/100$ over the choice of Y, A accepts.

The proof of the data milestone, as well as the first item of the lemma uses the same rationale of approximation by a Chernoff bound, and follows along the same lines. □

This concludes the proof that the milestones lemma implies Theorem 6.5. We have seen that A satisfies the third condition of a milestone function. The combination of Lemma 7.7, Lemma 7.8 and Lemma 7.10 gives that A also satisfies the first two conditions.

7.2.4 Proof of Lemma 7.7

We will prove the lemma in two steps that correspond to the two steps of the decoding: decoding control, and decoding data.

Claim 7.11. *For every* $m, s = (s_{\text{samp}}, s_\pi, s_{\text{PRG}}), r, e$ *and* y*, let* $c = \text{Enc}(m, s, r)$ *and* $c' = c \oplus e$*. If* $A_2^{weak}(m, s_{\text{samp}}, s_\pi, e, y) = 1$ *then* $\Pr[s \in \text{List}_{\text{ctrl}}] \geq 1 - \nu/20$ *(here the probability is over the coin tosses of the decoder, and* $\text{List}_{\text{ctrl}}$ *is the list obtained in the decoding algorithm described in Figure 3).*

Proof. By definition if $A_{\text{ctrl}}^{weak}(e) = 1$ iff *blocks*$(p+\epsilon) \leq (1 - \frac{\epsilon}{16}) \cdot n_{\text{ctrl}}$. This means that there are $\epsilon \cdot n_{\text{ctrl}}/16$ indices in $I = \text{Samp}(s_{\text{samp}})$ such that $\text{Dec}_{\text{ctrl}}(c'_i)$ decodes correctly (as only $(p + \epsilon) \cdot b$ errors were placed on the i'th block, and Dec_{ctrl} list-decodes from this number of errors). For every such i, the correct message s appears in the list $\text{Dec}_{\text{ctrl}}(c'_i)$.

In the first step, the decoding algorithm choose n' random indices from $[n]$. The fraction of good indices in $[n]$ is $\frac{\epsilon n_{\text{ctrl}}/16}{n} = \frac{\epsilon^2 n}{48}$. It follows (by a Chernoff bound) that with probability at least $1 - 2^{-\Omega(\epsilon^4 \cdot n')}$ at least $\epsilon^2 \cdot n'/100$ of the n' chosen indices $i_1, \ldots, i_{n'}$, have $s \in List_i$. Therefore, if this event occurs, s passes the pruning step and is selected to $\text{List}_{\text{ctrl}}$. Finally, we note that $1 - 2^{-\Omega(\epsilon^4 \cdot n')} \geq 1 - \nu/20$ by the choice of $n' = (\log N)^{c_\nu + 2}$. □

Claim 7.12. *For every* $m, s = (s_{\text{samp}}, s_\pi, s_{\text{PRG}}), r, e$ *and* y*, let* $c = \text{Enc}(m, s, r)$ *and* $c' = c \oplus e$*. If* $A_{\text{data}}^{weak}(m, s_{\text{samp}}, s_\pi, e, y) = 1$ *and* $s \in \text{List}_{\text{ctrl}}$ *(meaning that* s *was recovered correctly by the first step of decoding) then* $m \in \text{Dec}(c')$.

Proof. We have that $s \in \text{List}_{\text{ctrl}}$, meaning that s is one of the candidates considered in the second step of the decoding. Let y' be the string obtained from c' after the decoding uses s_{samp} to find the data blocks, s_{PRG} to unmask the data, and s_π to permute it back to it's original state. The requirement that $A^{weak}_{\text{data}}(m, s_{\text{samp}}, s_\pi, e, y) = 1$ implies that $A^m_w(e^\pi_{\text{data}}) = 1$. This means that the function $A^m(e^\pi_{\text{data}})$ from Theorem 3.20 answers one. By Theorem 3.20 this gives that

$$m = \text{Dec}_{\text{BSC}}(\text{Enc}_{\text{BSC}}(m) \oplus e^\pi_{\text{data}}).$$

Note that e^π_{data} is the error vector used on $\text{Enc}_{\text{BSC}}(m)$ in c'. More precisely, when decoding data in Figure 3 the string $x' = \text{Enc}_{\text{BSC}}(m) \oplus e^\pi_{\text{data}}$. It follows that m is in the final list of $\text{Dec}(c')$. ☐

The lemma follows from the combination of both claims.

7.2.5 Proof of Lemma 7.8

A good intuition to keep in mind is that we are trying to bound the harm that can be caused by an additive channel that uses fixed error vector e of Hamming weight at most pN.

We start by showing that with high probability, no more than an $\epsilon^2/4$ fraction of the control blocks, suffer too many errors from the error vector e.

Claim 7.13. *For every m, e of Hamming weight at most pn, y, and s_π,*

$$\Pr[A^{strong}_{\text{ctrl}}(m, S_{\text{samp}}, s_\pi, e, y) = 1] \geq 1 - \nu/100.$$

Proof. For a given error vector e we define:

$$T_e = \left\{i : \text{The } i\text{'th block has Hamming weight at most } (p + \tfrac{\epsilon}{4}) \cdot b\right\}.$$

For every e that has Hamming weight at most pN, it holds that $|T_e| > \frac{\epsilon}{4} \cdot n$ (otherwise we would have more than pN errors). Define $f_e : [n] \to \{0, 1\}$ such that $f_e(i) = 1$ iff $i \in T_e$. By the properties of the sampler Samp,

$$\Pr_{(z_1,\ldots,z_{n_{\text{ctrl}}}) \leftarrow \text{Samp}(U_{\ell'})}\left[\frac{1}{n_{\text{ctrl}}} \cdot |\{i : z_i \in T_e\}| - \frac{|T_e|}{n}| > \frac{\alpha}{100}\right] \leq \frac{\nu}{N^3}.$$

Thus, if we choose S_{samp} uniformly and independently we get that with probability $1 - \frac{\nu}{N^3}$, we have that *blocks*$(p + \frac{\epsilon}{4}) \leq (1 - \frac{\epsilon}{4} + \frac{\alpha}{100}) \cdot n_{\text{ctrl}} \leq (1 - \frac{\epsilon}{3}) \cdot n_{\text{ctrl}}$, and $A^{strong}_{\text{ctrl}}(m, S_{\text{samp}}, s_\pi, e, y) = 1$ (where the last inequality uses that $\alpha \leq \epsilon$).

☐

We now show that the fraction of errors induced by e to the data part cannot be significantly larger than p.

Claim 7.14. *For every m, e of Hamming weight at most pN, y, and s_π,*

$$\Pr_{s_{\text{samp}} \leftarrow U_{\ell'}}\left[\textit{weight}(e^{\text{Samp}(s_{\text{samp}})}_{\text{data}}) \geq N_{\text{data}} \cdot (p + \frac{\alpha}{100})\right] \leq \frac{\nu}{N^3}.$$

Proof. For a given error vector e, we define $f_e : [n] \to [0, 1]$ such that $f_e(i) = w_i$, where w_i is the relative weight of ith block in e. By the definition of the sampler,

$$\Pr_{(z_1,\ldots,z_{n_{\text{ctrl}}}) \leftarrow \text{Samp}(U_{\ell'})}\left[|\frac{1}{n_{\text{ctrl}}} \sum_{i \in [n_{\text{ctrl}}]} f(z_i) - p| > \frac{\alpha}{100}\right] \leq \frac{\nu}{N^3}.$$

Thus with probability $1-\frac{\nu}{N^3}$, the number of errors induced to the control blocks is at least $N_{\mathrm{ctrl}}(p-\frac{\alpha}{100})$, which implies that the number of error induced to the data is less than $pN-N_{\mathrm{ctrl}}(p-\frac{\alpha}{100}))\leq N_{\mathrm{data}}(p+\frac{\alpha}{100})$ (where the last inequality follows because $N_{\mathrm{ctrl}}\leq N_{\mathrm{data}}$). The claim follows. $\square$

In order to prove Lemma 7.8 it is sufficient to show that:

Claim 7.15. *For every m, e of Hamming weight at most pn, y, and s_{samp}. If s_{samp} is not one of the bad strings considered in the events of the two previous claims, then*

$$\Pr[A^{strong}_{\mathrm{data}}(m,s_{\mathrm{samp}},S_\pi,e,y)=1]\geq 1-\nu/100.$$

Proof. Let s_{samp} be a sampler seed for which the none of the two events of the previous two claims hold. (It is important to note that these events do not involve S_π). We have that the hamming weight of $\hat{e}=e^{\mathrm{Samp}(s_{\mathrm{samp}})}_{\mathrm{data}}$ is at most $(p+\frac{\alpha}{100})\cdot N_{\mathrm{data}}$. The function $A^{strong}_{\mathrm{data}}$ checks whether the function A_m from Theorem 3.20 accepts $\pi_{S_\pi}(\hat{e})$. When choosing $\mathrm{Enc}_{\mathrm{BSC}}$ in Figure 1, we applied Theorem 3.20 with $p_{\mathrm{BSC}}=p+\alpha$ and block length N_{data}. The string $\hat{e}$ has hamming weight less than $(p+\alpha)\cdot N_{\mathrm{data}}$, and therefore, by Theorem 3.20,

$$\Pr[A^{strong}_{\mathrm{data}}(m,s_{\mathrm{samp}},S_\pi,e,y)=1]=\Pr[A_m(\pi_{S_\pi}(\hat{e})=1]\geq 1-2^{-\Omega(t)}\geq 1-\nu/100,$$

where the last inequality follows because $t=(\log N)^{c_\nu+2}$. $\square$

Lemma 7.8 follows from the three claims above by noticing that $\nu/N^3+\nu/N^3+\nu/100\leq\nu/20$.

8 Conclusion and Open Problems

A natural open problem is to improve the running time of encoding and decoding to linear time. We remark that the step of applying a permutation $\pi[n]\to[n]$ on all n inputs, takes at least time $O(n\log n)$ (just to write down the inputs and outputs) and this is an obvious bottleneck for the approach used in this paper.

Guruswami and Smith [GS16] showed that we cannot expect to have uniquely decodable stochastic codes for space $\log n$ channels if $p>\frac{1}{4}$. However, it is not known whether for $p<\frac{1}{4}$, uniquely decodable stochastic codes for bounded space channels are possible with rate $R>1-H(2p)$ that is larger than the Gilbert-Varshamov bound (or even with a rate that matches the Gilbert-Varshamov bound, and efficient encoding and decoding).

Acknowledgement

We are very grateful to Benny Applebaum, Michael Forbes, William Hoza, Shachar Lovett, Or Meir, Noga Ron-Zewi, Chris Umans and Emanuele Viola for helpful conversations and advice.

We are very grateful to the organizers of Dagstuhl seminar 18391 on “Algebraic Methods in Computational Complexity” held on September 23-28, 2018. Some progress on this project was inspired by talks and personal conversations at the seminar.

We are very grateful to anonymous referees for detailed and very helpful comments.

References

[BR94] Mihir Bellare and John Rompel. Randomness-efficient oblivious sampling. In *35th Annual Symposium on Foundations of Computer Science*, pages 276–287, 1994.

[Cop82] Don Coppersmith. Rapid multiplication of rectangular matrices. *SIAM J. Comput.*, 11(3):467–471, 1982.

[DHRS07] Y. Ding, D. Harnik, A. Rosen, and R. Shaltiel. Constant-round oblivious transfer in the bounded storage model. *J. Cryptology*, 20(2):165–202, 2007.

[DPW18] Stefan Dziembowski, Krzysztof Pietrzak, and Daniel Wichs. Non-malleable codes. *J. ACM*, 65(4):20:1–20:32, 2018.

[FK18] Michael A. Forbes and Zander Kelley. Pseudorandom generators for read-once branching programs, in any order. In *59th IEEE Annual Symposium on Foundations of Computer Science, FOCS*, pages 946–955, 2018.

[For65] G David Forney. *Concatenated codes.* PhD thesis, Massachusetts Institute of Technology, 1965.

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

[Gol97] Oded Goldreich. A sample of samplers - a computational perspective on sampling (survey). *Electronic Colloquium on Computational Complexity (ECCC)*, 4(20), 1997.

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

[GS96] A. Garcia and H. Stichtenoth. On the asymptotic behavior of some towers of function fields over finite fields. *Journal of Number Theory*, 61(2):248–273, 1996.

[GS99] V. Guruswami and M. Sudan. Improved decoding of reed-solomon and algebraic-geometry codes. *IEEE Transactions on Information Theory*, 45(6):1757–1767, 1999.

[GS16] Venkatesan Guruswami and Adam Smith. Optimal rate code constructions for computationally simple channels. *Journal of the ACM (JACM)*, 63(4):35, 2016.

[GX15] Venkatesan Guruswami and Patrick Xia. Polar codes: Speed of polarization and polynomial gap to capacity. *IEEE Transactions on Information Theory*, 61(1):3–16, 2015.

[HAU14] Seyed Hamed Hassani, Kasra Alishahi, and Rüdiger L Urbanke. Finite-length scaling for polar codes. *IEEE Transactions on Information Theory*, 60(10):5875–5898, 2014.

[HL11] Ishay Haviv and Michael Langberg. Beating the gilbert-varshamov bound for online channels. In *2011 IEEE International Symposium on Information Theory Proceedings, ISIT 2011, St. Petersburg, Russia, July 31 - August 5, 2011*, pages 1392–1396, 2011.

[HLV18] Elad Haramaty, Chin Ho Lee, and Emanuele Viola. Bounded independence plus noise fools products. *SIAM J. Comput.*, 47(2):493–523, 2018.

[Joh62] Selmer Johnson. A new upper bound for error-correcting codes. *IEEE Transactions on Information Theory*, 8(3):203–207, 1962.

[Jus72] Jørn Justesen. Class of constructive asymptotically good algebraic codes. *IEEE Transactions on Information Theory*, 18(5):652–656, 1972.

[KNR09] E. Kaplan, M. Naor, and O. Reingold. Derandomized constructions of k-wise (almost) independent permutations. *Algorithmica*, 55(1):113–133, 2009.

[Kop10] Swastik Kopparty. *Algebraic methods in randomness and pseudorandomness*. PhD thesis, Massachusetts Institute of Technology, 2010.

[KS07] Tali Kaufman and Madhu Sudan. Sparse random linear codes are locally decodable and testable. In *48th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2007), October 20-23, 2007, Providence, RI, USA, Proceedings*, pages 590–600. IEEE Computer Society, 2007.

[KS13] Swastik Kopparty and Shubhangi Saraf. Local list-decoding and testing of random linear codes from high error. *SIAM Journal on Computing*, 42(3):1302–1326, 2013.

[Lan04] Michael Langberg. Private codes or succinct random codes that are (almost) perfect. In *45th Symposium on Foundations of Computer Science (FOCS 2004)*, pages 325–334, 2004.

[Lip94] Richard J. Lipton. A new approach to information theory. In *11th Annual Symposium on Theoretical Aspects of Computer Science*, pages 699–708, 1994.

[LV17] Chin Ho Lee and Emanuele Viola. More on bounded independence plus noise: Pseudorandom generators for read-once polynomials. *Electronic Colloquium on Computational Complexity (ECCC)*, 24:167, 2017.

[Mor93] Carlos Moreno. *Algebraic curves over finite fields*. Number 97 in Cambridge Tracts in Mathematics Series. Cambridge University Press, 1993.

[MPSW10] Silvio Micali, Chris Peikert, Madhu Sudan, and David A. Wilson. Optimal error correction for computationally bounded noise. *IEEE Trans. Information Theory*, 56(11):5673–5680, 2010.

[MS77] Florence Jessie MacWilliams and Neil James Alexander Sloane. *The theory of error-correcting codes*, volume 16. Elsevier, 1977.

[Sch06] Wolfgang M Schmidt. *Equations over finite fields: an elementary approach*, volume 536. Springer, 2006.

[Shp09] Amir Shpilka. Constructions of low-degree and error-correcting epsilon-biased generators. *Computational Complexity*, 18(4):495–525, 2009.

[Smi07] Adam D. Smith. Scrambling adversarial errors using few random bits, optimal information reconciliation, and better private codes. In *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA*, pages 395–404, 2007.

[SS16] Ronen Shaltiel and Jad Silbak. Explicit list-decodable codes with optimal rate for computationally bounded channels. In *APPROX/RANDOM*, pages 45:1–45:38, 2016.

[SSS95] Jeanette P. Schmidt, Alan Siegel, and Aravind Srinivasan. Chernoff-hoeffding bounds for applications with limited independence. *SIAM J. Discrete Math.*, 8(2):223–250, 1995.

[Sud97] M. Sudan. Decoding of Reed Solomon codes beyond the error-correction bound. *Journal of Complexity*, 13, 1997.

[Tho83] Christian Thommesen. The existence of binary linear concatenated codes with reed - solomon outer codes which asymptotically meet the gilbert- varshamov bound. *IEEE Trans. Information Theory*, 29(6):850–853, 1983.

[Vad04] Salil P. Vadhan. Constructing locally computable extractors and cryptosystems in the bounded-storage model. *J. Cryptology*, 17(1):43–77, 2004.

[Wei48] André Weil. On some exponential sums. *Proceedings of the National Academy of Sciences of the United States of America*, 34(5):204, 1948.

[Wil14] Ryan Williams. Faster all-pairs shortest paths via circuit complexity. In *Symposium on Theory of Computing, STOC*, pages 664–673, 2014.

A Proof of Theorem 3.20

In this subsection we prove Theorem 3.20. We start by specifying the inner and outer codes stated in the theorem statement, and then prove their relevant properties. We will use the following construction by Guruswami and Indyk [GI05].

Theorem A.1 (Theorem 3 [GI05])**.** *For every $0 < R < 1$, and every sufficiently small $\epsilon > 0$, there exists an explicitly specified family of codes of rate R and relative distance at least $(1-R-\epsilon)$ over an alphabet of size $2^{O(\frac{1}{\epsilon^4}\cdot\frac{1}{R}\cdot\log(\frac{1}{\epsilon}))}$, such that codes from the family can be encoded in linear time and can also be (uniquely) decoded in linear time from a fraction e of errors and s of erasures provided $2e + s \le (1 - R - \epsilon)$.*

In the following, we use some of the codes previously considered by [Smi07] and[GS16], and make small changes to fit our framework.

Outer code: Let $\mathrm{Enc}_{\mathrm{out}} : \{0,1\}^{k_{\mathrm{out}}} \to (\{0,1\}^{\log q_{\mathrm{out}}})^{n_{\mathrm{out}}}$ be the code guaranteed by Theorem A.1. These codes are encodable and decodable in linear time, can have rate $R_{\mathrm{out}} \ge 1 - \epsilon/10$ and are able to recover from $\lambda_1 = \epsilon/200$ fraction of error, with an alphabet size $q_{\mathrm{out}} \le 2^{\epsilon^{-7}}$. Since we can combine symbols to increase the alphabet size without compromising the fraction of correctable symbol errors, we will assume that $\log(q_{\mathrm{out}}) \in \Theta(\epsilon^{-7})$.

Inner code: By Shannon's theorem there exists binary linear codes $\mathrm{Enc}_{\mathrm{in}} : \{0,1\}^{k_{\mathrm{in}}} \to \{0,1\}^{n_{\mathrm{in}}}$, with rate $R_{\mathrm{in}} \ge 1 - H(p) - \epsilon/10$ that are decodable from $\mathrm{BSC}_{p'}^{n_{\mathrm{in}}}$ errors with probability $2^{-\Theta(\epsilon^2 n_{\mathrm{in}})}$, for every $0 \le p' \le p$. Thus we can choose $\lambda_2 \in \Theta(\epsilon^2)$. For our application we require that,

$$2 \cdot 2^{-(\lambda_2/2)\cdot n_{\mathrm{in}}} \le \lambda_1/100 \tag{2}$$

Hence, we get the added constrain that $n_{\mathrm{in}} \ge \Theta\Big(\frac{\log(1/\lambda_1)}{\lambda_2}\Big)$, which we satisfy by our choice of parameters since $\lambda_1 = \epsilon/200$ and $n_{\mathrm{in}} > k_{\mathrm{in}} = \log(q_{\mathrm{out}})$. The code is constructed by an exhaustive search over all linear codes, where $\mathrm{Dec_{in}}$ simply implements a maximum likelihood decoding. Both the decoding and construction can be performed in time $2^{\mathrm{poly}(n_{\mathrm{in}})}$. For more details, the reader is referred to [GRS12] Chapter 13.

Note that the concatenated code $\mathrm{Enc} = \mathrm{Enc_{out}} \circ \mathrm{Enc_{in}} : \{0,1\}^{k_{\mathrm{out}}} \to \{0,1\}^n$ specified by the above inner and outer codes is well defined, with rate $R \geq 1 - H(p) - \epsilon$, as intended.

In order to prove the penultimate item in Theorem 3.20, we prove the following claim, for which we introduce the following notation: For $e \in \{0,1\}^n$ denote $D_e = \pi_{U_d}(e)$. For every $i \in [n_{\mathrm{out}}]$ we use D_e^j to denote $(D_e)_{(j-1)\cdot n_{\mathrm{in}}+1} \cdots (D_e)_{j\cdot n_{\mathrm{in}}}$. The penultimate item in Theorem 3.20 will follow from the claim below:

Claim A.2. *Let $t \leq n^{0.1}$, and let $\pi : \{0,1\}^d \times [n] \to [n]$ be a $(2^{-10\cdot t}, t)$-wise independent permutation. For every $j \in [n_{\mathrm{out}}]$, let $A_m^j \colon \{0,1\}^{n_{\mathrm{in}}} \to \{0,1\}$ be a function that on input $y \in \{0,1\}^{n_{\mathrm{in}}}$ outputs one iff,*

$$\mathrm{Dec_{in}}(\mathrm{Enc_{in}}(\mathrm{Enc_{out}}(m)_j) \oplus y) \neq \mathrm{Enc_{out}}(m)_j,$$

For every $e \in \{0,1\}^n$ of Hamming weight at most pn, and every set S of $t' \leq \lfloor t/n_{\mathrm{in}} \rfloor$ distinct indices $i_1, \cdots, i_{t'} \in [n_{\mathrm{out}}]$, it holds that,

$$\Pr[A_m^{i_1}(D_e^{i_1}) = \ldots = A_m^{i_{t'}}(D_e^{i_{t'}}) = 1] \leq (2^{(-\lambda_2/2)\cdot n_{\mathrm{in}}})^{t'}$$

The proof of Claim A.2 is given below, but first we use the claim to prove the penultimate item in Theorem 3.20. First note that by definition,

$$\Pr[A_m(\pi_{U_d}(e)) = 0] = \Pr\big[\sum_{j \in n_{\mathrm{out}}} A_m^j(D_e^j) \geq w/10\big] \tag{3}$$

Thus to prove the penultimate item in Theorem 3.20, we need to show that

$$\Pr[\sum_{j \in n_{\mathrm{out}}} A_m^j(D_e^j) \geq w/10] = \Pr[\sum_{j \in n_{\mathrm{out}}} A_m^j(D_e^j) \geq (\lambda_1/10)\cdot n_{\mathrm{out}}] \leq 2^{-\lambda_3 \cdot t}.$$

For every $j \in [n_{\mathrm{out}}]$, let X_j denote $A_m^j(D_e^j)$. By Claim A.2 for every $t' \leq \lfloor t/n_{\mathrm{in}} \rfloor$ distinct indices $i_1, \cdots, i_{t'} \in [n_{\mathrm{out}}]$, $\Pr[X_{i_1} = \ldots = X_{i_{t'}} = 1] \leq \mu^{t'}$ for $\mu = (2^{(-\lambda_2/2)\cdot n_{\mathrm{in}}})$. Finally, by using the tail bounds stated at Lemma 3.12, we get that

$$\Pr[\sum_{j \in n_{\mathrm{out}}} X_j \geq 2 \cdot \mu \cdot n_{\mathrm{out}}] \leq 2^{-\lambda_3 t}, \tag{4}$$

for a sufficiently small constant λ_3. This concludes the proof since by our choice of parameters $2 \cdot \mu \leq \lambda_1/10$, (see Equation 2 above).

Proof of Claim A.2. Recall that we want to show that for every set S of $t' \leq \lfloor t/n_{\mathrm{in}} \rfloor$ distinct indices $i_1, \cdots, i_{t'} \in [n_{\mathrm{out}}]$,

$$\Pr[A_m^{i_1}(D_e^{i_1}) = \ldots = A_m^{i_{t'}}(D_e^{i_{t'}}) = 1] \leq 2^{-(\lambda_2/2)\cdot t}.$$

Let D_e^S denote $D_e^{i_1} \circ \cdots \circ D_e^{i_{t'}}$. We will compare D_e^S to $\mathrm{BSC}_p^{(t'\cdot n_{\mathrm{in}})}$, and show that they are close. Let $A_m^S \colon (\{0,1\}^{n_{\mathrm{in}}})^{t'} \to \{0,1\}$ be a function defined as follows: On input $y \in (\{0,1\}^{n_{\mathrm{in}}})^{t'}$, such that $y^j = y_{(j-1)\cdot n_{\mathrm{in}}+1} \cdots y_{j\cdot n_{\mathrm{in}}}$, $A_m^S(y) = 1$ iff for every $i_j \in S$, $A_m^{i_j}(y^j) = 1$. Note that this means that,

$$\Pr[A_m^S(D_e^S) = 1] = \Pr[A_m^{i_1}(D_e^{i_1}) = \ldots = A_m^{i_{t'}}(D_e^{i_{t'}}) = 1] \tag{5}$$

Claim A.3. *For every $0 < p < \frac{1}{2}$, $e \in \{0,1\}^n$ with Hamming weight pn, and S of size t',*

$$\Pr[A_m^S(D_e^S) = 1] \leq 2 \cdot \Pr[A_m^S(\mathrm{BSC}_p^{(t' \cdot n_{\mathrm{in}})}) = 1] + 2^{-10t}.$$

Proof of Claim A.3. Let us first imagine the case where π does not have an additive error, that is, π is a $(0,t)$-wise independent permutation. By definition, D_e^S can be viewed as sampling $\ell = t' \cdot n_{\mathrm{in}}$ uniform positions in the error vector e without replacement. On the other hand, BSC_p^ℓ can be viewed as sampling ℓ uniform positions in e with replacement. Let E denote the event where BSC_p^ℓ has no collision (no position was sampled twice). By definition this means that, $D_e^S \equiv (\mathrm{BSC}_p^\ell) \mid E$. Thus, for every function $A_m^S \colon \{0,1\}^\ell \to \{0,1\}$,

$$\Pr[A_m^S(D_e^S) = 1] = \Pr[A_m^S(\mathrm{BSC}_p^\ell) = 1 \mid E] \leq \frac{\Pr[A_m^S(\mathrm{BSC}_p^\ell) = 1]}{\Pr[E]} \leq \frac{\Pr[A_m^S(\mathrm{BSC}_p^\ell) = 1]}{1 - \binom{\ell}{2}/n} \tag{6}$$

Since π is only (2^{-10t})-close to a t-wise permutation, we get that,

$$\Pr[A_m^S(D_e^S) = 1] \leq 2 \cdot \Pr[A_m^S(\mathrm{BSC}_p^\ell) = 1] + 2^{-10t} \tag{7}$$

□

Note that by the properties of the inner code we specified in our construction, it holds that for every $0 \leq p' \leq p$ and $i_j \in S$: $\Pr[A_m^{i_j}(\mathrm{BSC}_{p'}^{n_{\mathrm{in}}}) = 1] \leq 2^{-\lambda_2 \cdot n_{\mathrm{in}}}$, and ultimately

$$\Pr[A_m^S(\mathrm{BSC}_{p'}^{(t' \cdot n_{\mathrm{in}})}) = 1] \leq (2^{-\lambda_2 \cdot n_{\mathrm{in}}})^{t'}.$$

Thus, Claim A.3 yields that for every S and $e \in \{0,1\}^n$ with Hamming weight at most pn,

$$\Pr[A_m^S(D_e^S) = 1] \leq 2 \cdot (2^{-\lambda_2 \cdot n_{\mathrm{in}}})^{t'} + 2^{-10t} \leq (2^{(-\lambda_2/2) \cdot n_{\mathrm{in}}})^{t'}$$

□